MASTER'S DEGREE THESIS

Master of Science in Advanced Mathematics and Mathematical Engineering

Title: New lattice-based protocols for proving correctness of a shuffle

Author: Manuel Sánchez Torrón

Advisor: Javier Herranz Sotoca and Ramiro Martínez Pinilla

Department: Mathematics

Academic year: 2019-2020

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master in Advanced Mathematics and Mathematical Engineering

Master's thesis

New lattice-based protocols for proving correctness of a shuffle

Manuel Sánchez Torrón

Supervised by Javier Herranz Sotoca and Ramiro Martínez Pinilla

June, 2020

Thanks to my advisors Javier Herranz and Ramiro Martínez for their invaluable help in this thesis and for teaching me everything I know about cryptography.

Thanks to my family for their unconditional support throughout these years.

Thanks to all my friends for always being there, in the easy moments and in the tough ones. Special thanks to my brothers from different parents, Juan and Pablo.

And above all, many thanks to my parents, for giving me everything I have, teaching me (almost) everything I know, and setting me on the path to one day becoming the man I aspire to be.

Moitas grazas a todos.

Abstract

In an electronic voting procedure, mixing networks are used to ensure anonymity of the casted votes. Each node of the network re-encrypts the input and randomly permutes it in a process named shuffle, and must prove that the process was applied honestly. State-of-the-art classical proofs achieve logarithmic communication complexity on N (the number of votes to be shuffled) but they are based on assumptions which are weak against quantum computers. To maintain security in a post-quantum scenario, new proofs are based on different mathematical assumptions, such as lattice-based problems. Nonetheless, the best lattice-based protocols to ensure verifiable shuffling have linear communication complexity on N. In this thesis we propose the first sub-linear post-quantum proof for the correctness of a shuffle, for which we have mainly used two ideas: arithmetic circuit satisfiability results from [BBC+18] and Beneš networks to model a permutation of N elements. The achieved complexity of our protocol is $\mathcal{O}(\sqrt{N}\log^2(N))$.

Keywords

electronic voting, lattice-based cryptography, RLWE-encryption, zero-knowledge proofs, arithmetic circuit satisfiability, Beneš permutation networks, proof of a shuffle.

Contents

1 Introduction **4**

1.1 Computational complexity 4

1.2 Quantum computing 6

1.3 Outline of the thesis 7

2 Preliminaries **8**

3 Lattices **10**

3.1 Basic concepts 10

3.2 Lattice-based problems 12

3.3 Ideal lattices 14

3.4 Ring learning with errors 15

4 Public-key cryptography **17**

4.1 Public-key encryption 17

4.1.1 RLWE encryption scheme 19

4.2 Trapdoor functions 20

4.2.1 Ajtai's One-Way function 21

4.3 Commitment schemes 21

4.3.1 A SIS-based commitment scheme 22

4.4 Zero-Knowledge proofs of knowledge 23

4.4.1 Zero-knowledge argument for preimages of Ajtai's function 26

5 Arithmetic circuit satisfiability **28**

5.1 Introduction to arithmetic circuits 28

5.2 Reduction to polynomial equations 29

5.3 Arithmetic circuit satisfiability argument 31

5.3.1 Argument for multiplicative relations and linear constraints 32
5.3.2 Efficiency 33
5.4 Example: RLWE encryption proof 33

6 Proof of a shuffle **36**
6.1 Voting process 36
6.2 The random permutation: Beneš networks 38
6.3 A new lattice-based protocol for the correctness of a shuffle 42

7 Conclusions **44**

A Argument for multiplicative relations **48**

B Argument for linear consistency constraints **52**

1. Introduction

In a society like the current one, electronic procedures are becoming more numerous, important and sophisticated. Online shopping, bank management, or even watching our favourite series in a video on demand streaming service are activities that today can be carried out almost at any time and anywhere with our mobile phone. All these activities have something in common: we wish to keep our personal information private. One of these processes is electronic voting, which is becoming more and more a reality. This kind of voting could bring several advantages compared to traditional voting. Electronic voting is less expensive, it requires of less infrastructure, the results are obtained quicker and the process is way much simpler. However, to become a real alternative, it requires to be as reliable as standard voting. Hence the importance of cryptography, since it may allow security, privacy and integrity to be guaranteed in these processes.

The purpose of cryptography is to permit confidential communication between two parties through an insecure channel. That is, it allows to send a message in such a way that everyone can read it, but only the involved people can understand it. To achieve this, it uses mathematical problems that are difficult to solve unless some piece of information is known. Initially, for two parties to communicate it was necessary that they previously agreed on a secret key, which gave rise to private key cryptography. This cryptographic paradigm involved many difficulties, principally regarding the necessity of large keys. In 1976 Whitfield Diffie and Martin Hellman created a protocol to share information without the need of the previous agreement on a secret key [DH76]. In this way, they made possible the so-called public key cryptography, in which each party has a private key and a public key, and that solves many of the issues that private key cryptography had.

The new paradigm of programming based on quantum physics has great potential in terms of computational power, and therefore it might suppose a threat to current cryptography, since some problems which remained almost insolvable for years may become not so difficult for a quantum computer. The objective of this thesis is to present a new protocol useful in electronic voting, and prove its security against a quantum adversary.

1.1 Computational complexity

When studying the various problems it is especially interesting to know the cost of resources of a computer such as time and memory space that are required to run algorithms that solve each problem. Specifically, computational complexity theory aims to make a classification of the problems according to the inherent difficulty of each problem, as well as to discern which problems can be solved by a computer and which cannot.

The complexity of an algorithm usually varies according to the size n of the input. Let

$n \mapsto f(n)$ be the positive function that maps the size of the input to the maximum (or average) amount of resources needed to solve the problem over all possible n. Then f is called the worst-case (or respectively, average-case) complexity of the algorithm.

For theoretical purposes, it is specially relevant the asymptotic behaviour of the function when n goes to infinity. This behaviour is usually expressed using the big-O notation. We say that $f(n) \in \mathcal{O}(g(n))$ if there exists $\varepsilon > 0$ and n_0 such that for all $n \geq n_0$: $f(n) \leq \varepsilon \cdot g(n)$. Informally, this means that g grows at least at the same speed than f. For instance, if an algorithm runs in $f(n) = 4n^3 + n + \log(n)$, as n grows, the n^3 term dominates the others and the constant 4 becomes irrelevant. Therefore, we have $f(n) \in \mathcal{O}(n^3)$. This notation allows to classify algorithms according to their complexity, choosing functions to be the representative of the different asymptotic behaviours. Table 1 shows some of the complexity classes that will appear throughout the thesis, in strict increasing order by set inclusion.

Notation	Name	Other information
$\mathcal{O}(1)$	Constant	
$\mathcal{O}((\log(n))^c)$	Polylogarithmic	For $c > 0$
$\mathcal{O}(n^c)$	Fractional power	For $0 < c < 1$
$\mathcal{O}(n)$	Linear	
$\mathcal{O}(n \log(n))$	Quasilinear	
$\mathcal{O}(n^c)$	Polynomial	For $c > 1$. Also denoted poly(n)
$\mathcal{O}(c^n)$	Exponential	For $c > 1$

Table 1: Complexity classes.

It is also commonly used the big-Omega notation, which represents the opposite of the big-O. A function $f(n) \in \Omega(g(n))$ if and only if $g(n) \in \mathcal{O}(f(n))$. In addition, it is possible to define $\Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$.

It is important to distinguish between the complexity of an algorithm and the complexity of a problem. The complexity of a problem refers to its inherent difficulty, which may not be known. In computer science, by convention, a problem is considered easy to solve if there is knowledge of a polynomial-time algorithm that solves it.

Hence it is possible to classify problems according to their complexity. There are several problem classes, but the most important ones are classes $\mathcal{P}$ and $\mathcal{NP}$. A common misconception is to consider that $\mathcal{P}$ are the problems solved by a polynomial-time algorithm, while $\mathcal{NP}$ are the ones whose best solution is exponential or worse. This is not exact, although is close. Actually, the names $\mathcal{P}$ and $\mathcal{NP}$ refer to *deterministic polynomial* and *non-deterministic polynomial*, respectively. This means that problems in $\mathcal{P}$ can be solved by a deterministic polynomial-time Turing machine, and problems in $\mathcal{NP}$ can be solved by a non-deterministic polynomial-time Turing machine. Informally, a deterministic Turing machine is equivalent to standard deterministic algorithms, while a non-deterministic Turing machine is a theoretical construction in which algorithms are allowed to execute some instructions at the same time.

While it is obvious that $\mathcal{P} \subseteq \mathcal{NP}$, since deterministic machines are just a kind of non-deterministic machines; a much more intriguing question, and one of the most important problems in mathematics and computer science, is whether $\mathcal{NP} \subseteq \mathcal{P}$. There is a sub-class of problems contained in $\mathcal{NP}$ called $\mathcal{NP}$-complete, which has the particularity that finding just one deterministic polynomial time solution for just one of these problems proves $\mathcal{P} = \mathcal{NP}$. Nonetheless, this algorithm, if it exists, at the moment is not known and the problem $\mathcal{P}$ vs. $\mathcal{NP}$ remains unsolved.

In cryptography it is very unusual to mathematically prove absolute security of applications. Instead, the most common solution to this issue is to base security on the inherent hardness of certain problems and assumptions such as $\mathcal{P} \neq \mathcal{NP}$.

1.2 Quantum computing

The basis of classic computing are bits, which may have a value of 0 or 1. Physically, they are usually implemented using high or low electric potential on a wire. However, there are other ways to implement the information of a bit. We can consider an hydrogen atom, for example, and check if its electron is at its lowest energy level (fundamental state) or if its at a higher level of energy (excited state), and encode each state with 0 or 1. We will use the bra-ket notation $|0\rangle$ and $|1\rangle$ to represent these values. When considering this type of information encoding, the quantum properties appear and they must be taken into account.

What makes quantum computation different from classical computation is that the electron is in a linear superposition of both states, namely in a quantum state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. This state $|\psi\rangle$ is called a *qubit* and it is the basis of quantum computing. However, to read the value of a qubit we must perform a *measurement*, which will return $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$, and will cause the qubit to collapse to the corresponding classical state.

We can consider states with more than one qubit, for instance the 2-qubit state $|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle$. If we have two qubits $\alpha\,|0\rangle + \beta\,|1\rangle$ and $\gamma\,|0\rangle + \delta\,|1\rangle$ we can consider the 2-qubit state formed by those 2 qubits using the tensor product to obtain $|\psi\rangle = \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle$. This implies another important feature of quantum physics, which is called *entanglement*. Notice that there are 2-qubit states such as $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ that cannot be written as a product of two single states. Quantum entanglement might have some really powerful applications in telecommunications and coding theory, since two entangled qubits could be used to share information instantaneously between them even if they are separated by millions of light years.

For our purposes, the relevance of quantum algorithms is that they are able to solve some $\mathcal{NP}$ problems which are not known to be in $\mathcal{P}$ in polynomial time. Some of the most used cryptographic applications have hypothetical security that relies on the hardness of problems of this kind, such as the discrete logarithm or the factorization problem, which can be solved

with Shor's algorithm [Sho97]. Therefore, some cryptographic protocols may become insecure in front of a quantum computer. At the present time no one has access to a sufficiently powerful quantum computer to obtain encrypted data, but it is necessary to adapt as soon as possible our cryptographic protocols to avoid these potential attacks, in order to achieve long-term privacy. In fact, a malicious adversary might store the encrypted information until it has access to a quantum computer and then proceed to recover this private data.

There are classical problems that are hard even for quantum computers, in the sense that there is not knowledge of any quantum algorithm capable of solving any of them in polynomial time. Usually, a quantum algorithm is able to solve this problems faster than classical algorithms but still in exponential time Therefore, the security parameters should be established taking quantum computing into account, to keep security. Post-quantum cryptography is the branch that studies how to develop new applications based on these problems in order to achieve security against quantum computers. One of the most promising families of post-quantum problems are based in a mathematical construction called lattices. In this thesis we will explain the main results about lattices and the problems derived from them which are hypothetically resistant against quantum adversaries.

1.3 Outline of the thesis

In section 1 we introduce the general purpose of cryptography, we explain the theory of computational complexity and we introduce the new paradigm based on quantum physics. In section 2 we explain the notation used throughout the thesis. In section 3 we illustrate the mathematical background of lattices and explain the main problems which are used in cryptographic protocols given their assumed hardness against quantum computers. In section 4 we introduce the main cryptographic primitives such as public key encryption, commitment schemes and zero-knowledge proofs with some useful examples. In section 5 we introduce arithmetic circuits and show how to prove satisfiability in zero knowledge. In section 6 we define a shuffle and we combine the previous sections to achieve the main objective of the thesis: a new lattice-based zero knowledge argument for the correctness of a shuffle using arithmetic circuit satisfiability. Finally, in section 7 we summarize the whole thesis and discuss some possibilities for future work.

2. Preliminaries

We denote vectors with boldface lower-case letters like $\mathbf{a}$, matrices with boldface upper-case letters like $\mathbf{A}$. We represent $\mathbf{A} = [\mathbf{a_1}, \dots, \mathbf{a_k}]$ the matrix whose i-th column is the vector $\mathbf{a_i}$. The euclidean norm of an element a is $\|a\|_2 = |a|$ if $a \in \mathbb{Z}$ or $\|a\|_2 = \sqrt{\sum a_i^2}$ if $a = \sum a_i X^i \in \mathbb{Z}[X]$. We extend the euclidean norm notation to vectors $\|\mathbf{a}\|_2 = \sqrt{\sum \|a_i\|_2^2}$ and matrices $\|\mathbf{A}\|_2 = \sqrt{\sum \|\mathbf{a_i}\|_2^2}$. The standard inner product of two vectors $\mathbf{a}$, $\mathbf{b}$ is $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i b_i$. We also consider the operator norm of matrices defined over rings $s_1(\mathbf{A}) = \max_{\mathbf{x} \neq 0} \left(\frac{\|\mathbf{Ax}\|_2}{\|\mathbf{x}\|_2} \right)$, the largest eigenvalue of matrix $\mathbf{A}$.

We write $x \xleftarrow{\mathcal{R}} X$ when x is chosen uniformly at random from a set X, and $x \xleftarrow{\mathcal{R}} \chi$ when x is chosen according to the probability distribution χ. Given a deterministic algorithm $\mathcal{A}$ we write $y \leftarrow \mathcal{A}(x)$ when $\mathcal{A}$ outputs y on input x. Similarly, if $\mathcal{A}$ is probabilistic, we write $y \leftarrow \mathcal{A}(x; r)$ when $\mathcal{A}$ outputs y on input x and randomness r, or $y \xleftarrow{\mathcal{R}} \mathcal{A}(x)$, omitting the explicit randomness for the sake of clarity. Henceforward, we refer to Probabilistic Polynomial Time algorithm as PPT algorithm.

Given the function $\rho_\sigma(x) = e^{\frac{-x^2}{2\sigma^2}}$, the discrete Gaussian distribution over the integers D_σ is:

$$D_\sigma(x) = \frac{\rho_\sigma(x)}{\rho_\sigma(\mathbb{Z})} \quad \text{where } \rho_\sigma(\mathbb{Z}) = \sum_{v \in \mathbb{Z}} \rho_\sigma(v)$$

We can upper-bound the norm of $x \xleftarrow{\mathcal{R}} D_\sigma$ for every $\sigma > 0$ [Ban93], since for every $k > 0$ it holds that:

$$\Pr\left[|x| > k\sigma \mid x \xleftarrow{\mathcal{R}} D_\sigma \right] \leq 2e^{-k^2/2}$$

and when $\mathbf{x} \xleftarrow{\mathcal{R}} D_\sigma^n$ it holds:

$$\Pr\left[\|\mathbf{x}\|_2 > \sqrt{2n}\sigma \mid \mathbf{x} \xleftarrow{\mathcal{R}} D_\sigma^n \right] < 2^{-n/4}$$

Definition 2.1 (Negligible function)**.** We call a non-negative function $f : \mathbb{Z}^+ \to \mathbb{R}$ negligible if it vanishes faster than the inverse of any nonzero polynomial:

$$\forall c \in \mathbb{Z}^+ \quad \lim_{\lambda \to \infty} f(\lambda) \lambda^{-c} = 0$$

We denote the set of negligible functions over the variable λ as $\mathrm{negl}(\lambda)$.

Definition 2.2 (Overwhelming function)**.** We say a function f is overwhelming if

$$1 - f \in \mathrm{negl}(\lambda)$$

Whenever defining a cryptographic scheme, it will require a security parameter λ, which will indicate the security of the scheme. In particular, we will ask for some probabilities to be negligible over λ, thus when λ increases, the restrictions become stronger, and therefore the sheme more secure. For this reason we will denote 1^λ as the unary representation of λ. The reason for taking the unary representation is to avoid using logarithms when doing calculations with the size of λ, since the size of 1^λ is precisely λ. For instance, if $\lambda = 5$ then $1^\lambda = 11111$.

3. Lattices

3.1 Basic concepts

Definition 3.1 (Lattice)**.** A lattice $\mathcal{L}$ is defined as a set of points in an n-dimensional space (usually $\mathbb{R}^n$) that verify these two conditions:

1. It is an **additive subgroup**:
   - $0 \in \mathcal{L}$
   - $\forall x, y \in \mathcal{L} : \ -x, x + y \in \mathcal{L}$.
2. It is **discrete**: for every $x \in \mathcal{L}$, there is a neighbourhood of x in the base space such that x is the only point in $\mathcal{L}$.

In other words, a lattice is a set of points in an n-space with a periodic structure.

Henceforward we will always consider the real space $\mathbb{R}^n$ as the base space.

Lattices can be seen as vector spaces generated by a basis with the restriction that the coefficients must be integers, instead of real values. Thus, we may consider lattices generated by a basis in an analogous way as in usual vector spaces.

Definition 3.2 (Generated lattice)**.** Given a set of k linearly independent vectors $\{\mathbf{b_1}, \dots, \mathbf{b_k}\} \subset \mathbb{R}^n$, the generated lattice is:

$$\mathcal{L}(\mathbf{b_1}, \dots, \mathbf{b_k}) = \left\{ \sum_{i=1}^{k} x_i \mathbf{b_i} : \ x_i \in \mathbb{Z}, \text{ for } 1 \leq i \leq k \right\} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\} = \mathcal{L}(\mathbf{B})$$

Where we have used the matrix notation $\mathbf{B} = [\mathbf{b_1}, \dots, \mathbf{b_k}] \in \mathbb{R}^{k \times n}$. We will say that the lattice $\mathcal{L}(\mathbf{B})$ is an n-dimensional lattice of rank k generated by the basis $\{\mathbf{b_1}, \dots, \mathbf{b_k}\}$. When $k = n$ the lattice is called a full-rank lattice. If not stated otherwise, we will only consider full-rank lattices.

The basis of a lattice is not unique. In fact, every lattice can be generated by infinitely many bases, and some of them may be better than others. In particular, if we consider an unimodular matrix $\mathbf{U}$ (that is, an integer square matrix with determinant 1 or -1), then the bases $\mathbf{B}$ and $\mathbf{BU}$ generate the same lattice.

Theorem 3.3. $\mathcal{L}(\mathbf{B_1}) = \mathcal{L}(\mathbf{B_2})$ *if and only if* $\mathbf{B_2} = \mathbf{B_1 U}$ *for some unimodular matrix* $\mathbf{U}$.

Proof. Suppose $\mathcal{L}(\mathbf{B_1}) = \mathcal{L}(\mathbf{B_2})$. Then each of the columns $\mathbf{b_i}$ of $\mathbf{B_2}$ is in $\mathcal{L}(\mathbf{B}_1)$. Hence, exists an integer matrix $\mathbf{U} \in \mathbb{Z}^{n\times n}$ such that $\mathbf{B_2} = \mathbf{B_1U}$. Analogously, there is $\mathbf{V} \in \mathbb{Z}^{n\times n}$ such that $\mathbf{B_1} = \mathbf{B_2V}$. Therefore $\mathbf{B_2} = \mathbf{B_2VU}$ and we get, taking determinants:

$$\det(\mathbf{B_2}) = \det(\mathbf{B_2VU}) = \det(\mathbf{B_2})\det(\mathbf{V})\det(\mathbf{U})$$

Therefore $\det(\mathbf{U})\det(\mathbf{V}) = 1$. And given that $\mathbf{U}$ and $\mathbf{V}$ are integer matrices, we have $\det(\mathbf{U}) = \pm 1$.

Now suppose $\mathbf{B_2} = \mathbf{B_1U}$, for some unimodular matrix $\mathbf{U}$. Then, each column $\mathbf{b_i}$ of $\mathbf{B_2}$ is contained in $\mathcal{L}(\mathbf{B_1})$, so $\mathcal{L}(\mathbf{B_1}) \subseteq \mathcal{L}(\mathbf{B_2})$. Also, since $\mathbf{U}^{-1}$ is also unimodular and $\mathbf{B_1} = \mathbf{B_2U}^{-1}$, we get $\mathcal{L}(\mathbf{B_2}) \subseteq \mathcal{L}(\mathbf{B_1})$. In conclusion $\mathcal{L}(\mathbf{B_1}) = \mathcal{L}(\mathbf{B_2})$.

□

Notice that this implies that the absolute value of the determinant of a basis of a lattice is an invariant of the lattice. Therefore we may define the following concept:

Definition 3.4 (Determinant of a lattice)**.** The determinant of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is:

$$\Delta(\mathcal{L}) = |\det(\mathbf{B})|$$

Now that it is clear that the same lattice can be generated by several bases, let us formally state whether a basis is a 'good' one or a 'bad' one. We are interested in 'small' bases with high 'orthogonality', which allow some cryptographic applications to have a higher chance of success. For this purpose, we introduce the concept of orthogonality defect.

Definition 3.5 (Orthogonality defect)**.** Given a lattice $\mathcal{L}(\mathbf{B})$, the orthogonality defect is:

$$\delta(\mathbf{B}) = \frac{\prod_{i=1}^{n} \|\mathbf{b_i}\|_2}{\det(\mathbf{B})}$$

It is possible to consider this value normalized by taking the n-root: $\sqrt[n]{\delta(\mathbf{B})}$

The orthogonality defect is 1 if and only if the basis $\mathbf{B}$ is orthogonal. If not, the defect increases as the orthogonality of the basis decreases. However, given a bad basis of a lattice, transforming it into a good basis is in general a difficult task. When $n = 2$ a gaussian elimination provides a good basis and this method can be generalized to a n-dimensional lattice using the Lenstra-Lenstra-Lovász algorithm presented in [LLL82], which applies gaussian elimination to the elements of the basis two by two. Nevertheless, the length of the vectors of the basis are far from optimal.

Several problems about lattices that will be used for cryptographic purposes involve the search of small elements in a lattice. Hence, we introduce the concept of minimum of a lattice.

Definition 3.6 (Minimum of a lattice)**.** The minimum of a lattice $\lambda_i(\mathcal{L})$ is the radius of the smallest n-sphere centered in the origin containing at least i linearly independent vectors of the lattice. In particular, $\lambda_1(\mathcal{L})$ is the minimum distance between two elements of the lattice, which is equivalent to the length of the shortest non-zero vector:

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \{0\}} \|\mathbf{v}\|_2$$

Definition 3.7 (Dual of a lattice)**.** Given a lattice $\mathcal{L}$ the dual lattice is defined as:

$$\mathcal{L}^* = \{y \in \mathbb{R}^n \mid \langle x, y \rangle \in \mathbb{Z}\}$$

It can be seen that if $\mathbf{B}$ is a basis of a lattice $\mathcal{L}$, then $(\mathbf{B}^{-1})^\mathsf{T}$ is a basis of the dual lattice $\mathcal{L}^*$

To make computers able to work with lattices, it is necessary to consider lattices modulo an integer q. Hence, we introduce the concept of q-ary lattices.

Definition 3.8 (q-ary lattice)**.** Given a (possibly prime) integer q, a lattice $\mathcal{L}$ is said to be q-ary if it satisfies:

$$q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$$

That is, the membership of $x \in \mathbb{R}^n$ in $\mathcal{L}$ is determined by $x \mod q$. When $\mathcal{L} \subset \mathbb{Z}^n$ and q is a multiple of $\Delta(\mathcal{L})$, the lattice $\mathcal{L}$ is a q-ary lattice. However, interesting q-ary lattices for cryptography involve a q much smaller than the determinant of the lattice.

There are two ways to define q-ary lattices. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the rows of $\mathbf{A}$ generate the lattice

$$\Lambda_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^n \mid \mathbf{y} = \mathbf{A}^\mathsf{T}\mathbf{s} \mod q \text{ for some } \mathbf{s} \in \mathbb{Z}^m\}$$

while the elements which are orthogonal modulo q to the rows of $\mathbf{A}$ generate the orthogonal lattice

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{y} = 0 \mod q\}$$

It is important to notice that the matrix $\mathbf{A}$ is not a basis of these lattices.

And these lattices are dual of each other, up to normalization; namely $\Lambda_q^\perp(\mathbf{A}) = q\lambda_q(\mathbf{A})^*$ and $\Lambda_q(\mathbf{A}) = q\Lambda_q^\perp(\mathbf{A})^*$.

3.2 Lattice-based problems

Now we proceed to illustrate the main problems based in lattices. The assumed hardness of these problems against quantum computers will be the key to ultimately design highly secure cryptographic schemes, in principle capable to resist even quantum attacks.

Definition 3.9 (Approximate Shortest Vector Problem (γ-SVP)**.** Given a basis $\mathbf{B}$ of a lattice $\mathcal{L}(\mathbf{B})$ the *Approximate Shortest Vector Problem* γ-SVP, is to find a non-zero vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that

$$\|\mathbf{v}\|_2 \leq \gamma(n) \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$$

The parameter γ is typically taken to be a function of the lattice dimension n, and is the key of the hardness of γ-SVP. Observe the problem gets harder as γ gets smaller. When $\gamma = 1$, the problem is called simply *Shortest Vector Problem*, and it has been proved to be NP-hard. In the approximate case, some polynomial algorithms for basis reductions like the L^3 [LLL82] can find a solution when γ is large enough ($\gamma \approx 2^{\Omega(n)}$), and some descendants improved the algorithm to obtain solutions when $\gamma \approx 2^{\Theta(n \log\log n / \log n)}$. For $\gamma \approx \text{poly}(n)$ it is believed that no PPT can solve this problem, since the best algorithms known either require super-exponential $2^{\Theta(n \log n)}$ time or exponential $2^{\Theta(n)}$ time and space.

Definition 3.10 (Approximate Closest Vector Problem (γ-CVP)**.** Given a basis $\mathbf{B}$ of a lattice $\mathcal{L}(\mathbf{B})$ and a target vector $\mathbf{t} \in \mathbb{R}^n$ the *Approximate Closest Vector Problem* γ-CVP, is to find a vector $\mathbf{u} \in \mathcal{L}(\mathbf{B})$ such that if $\mathbf{v} = \arg\min_{\mathbf{w} \in \mathcal{L}(\mathbf{B})} \|\mathbf{t} - \mathbf{w}\|_2$ then

$$\|\mathbf{u} - \mathbf{v}\|_2 \leq \gamma(n) \|\mathbf{t} - \mathbf{v}\|_2$$

While the definition might look a bit confusing, the problem is just finding the vector $\mathbf{u} \in \mathcal{L}(\mathbf{B})$ which is closest to the target $\mathbf{t} \in \mathbb{R}^n$. Goldreich *et al.* proved in [GMSS99] that hardness of the γ-CVP is at least the same as γ-SVP.

Definition 3.11 (Approximate Shortest Independent Vectors Problem (γ-SIVP)**.** Given a basis $\mathbf{B}$ of a lattice $\mathcal{L}(\mathbf{B})$, find n linearly independent vectors $\mathbf{s}_i \in \mathcal{L}(\mathbf{B})$, where

$$\|s_i\|_2 \leq \gamma(n) \cdot \lambda_n(\mathcal{L}(\mathbf{B}))$$

for all i.

Now we proceed to define one of the main problems about lattices, the Learning With Errors Problem (LWE). This problem was first proposed by Regev in [Reg05] and it is in the core of many post-quantum public key encryption schemes.

Let n, q integers, χ_σ a discrete probability distribution over $\mathbb{Z}$ (usually a Gaussian distribution) with standard deviation σ and a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$.

Definition 3.12 (Learning With Errors Distribution)**.** The LWE distribution $\mathcal{L}_{\mathbf{s},\chi_\sigma}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $\mathbf{a} \xleftarrow{\mathcal{R}} \mathbb{Z}_q^n$, $e \xleftarrow{\mathcal{R}} \chi_\sigma$ and outputting $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle + e \mod q)$.

Definition 3.13 (Search-LWE Problem)**.** Given m independent samples $(\mathbf{a_i}, b_i) \xleftarrow{\mathcal{R}} \mathcal{L}_{\mathbf{s},\chi_\sigma}$ for a fixed uniformly random $\mathbf{s}$, find $\mathbf{s}$.

Definition 3.14 (Decision-LWE Problem)**.** Given m independent samples $(\mathbf{a_i}, b_i)$, decide whether this samples are distributed according to $\mathcal{L}_{\mathbf{s},\chi_\sigma}$ for a fixed uniformly random $\mathbf{s}$; or according to an uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The last problem we will explain is called the *Short Integer Solution*, proposed by Ajtai in [Ajt96]. This problem yields a collision-resistant hash function, and will be useful to define a lattice-based compressing commitment scheme in section 4.3.1.

Definition 3.15 (Short Integer Solution (SIS))**.** Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and a positive $\gamma \in \mathbb{R}$, the *Short Integer Solution* consists in finding a non-zero vector $\mathbf{z} \in \mathbb{Z}^m$ with $\|\mathbf{z}\|_2 \leq \gamma$ such that:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{Az} = \mathbf{0} \in \mathbb{Z}_q^n$$

Notice that this problem is equivalent to finding a sufficiently short non-zero vector in the lattice $\Lambda_q^{\perp}(\mathbf{A})$.

This problem also admits an inhomogeneous version, in which given a target vector $\mathbf{t} \in \mathbb{Z}_q^n$, the goal is to find a short integer solution to the equation $\mathbf{Az} = \mathbf{t}$, where $\mathbf{A}$ and $\mathbf{t}$ are independent and chosen uniformly at random.

These problems are supposed to be resistant against quantum adversaries, and therefore are used in many cryptographic applications in order to achieve long term security.

3.3 Ideal lattices

Encryption schemes based on these lattice problems are the main hope for post-quantum cryptography. Nonetheless, considering lattices as explained above arise some issues that make these schemes impractical. The most important one is that for an n-dimensional lattice basis, we require storage for an $n \times n$ matrix, and therefore quadratic space on the size of the lattice, which becomes big for a reasonable dimension n. To solve this issue, along with some other problems regarding speed of operations, ideal lattices were proposed.

Given a fixed vector $\mathbf{f} = (f_1, \dots, f_n) \in \mathbb{Z}^n$ we define the following transformation matrix:

$$\mathbf{F} = \left[\begin{array}{ccc|c} 0 & \dots & 0 & -f_1 \\ \ddots & & & -f_2 \\ & \mathbf{I}_{n-1} & & \vdots \\ & & \ddots & -f_n \end{array}\right]$$

Definition 3.16 (Ideal lattice)**.** An ideal lattice $\mathcal{L}$ is the lattice generated by the matrix

$$\mathbf{A} = \left[\mathbf{a}, \mathbf{Fa}, \dots, \mathbf{F}^{n-1}\mathbf{a}\right]$$

for a vector $\mathbf{a} \in \mathbb{Z}^n$.

The name *ideal lattices* comes from the fact that these kind of lattices can be seen as ideals in the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\langle f(X)\rangle$, where the polynomial $f(X) = X^n + f_n X^{n-1} + \cdots + f_2 X + f_1 \in \mathbb{Z}[X]$. Usually, for real cryptographic applications, we will set n a power of 2 and $f(X) = X^n + 1$ and we will consider the quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ since this setting provides several advantages from an implementation point of view (see section 4.1.1).

In this case the matrix $\mathbf{A}$ is an anti-cyclic integer matrix:

$$\mathbf{A} = \begin{bmatrix} a_1 & -a_n & -a_{n-1} & \dots & -a_2 \\ a_2 & a_1 & -a_n & \dots & -a_3 \\ a_3 & a_2 & a_1 & \dots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{bmatrix}$$

Henceforth we will consider elements in $\mathcal{R}_q$ either as a vector $(a_1, \dots, a_n) \in \mathbb{Z}_q^n$ or as the polynomial $a = \sum_{i=1}^{n} a_i X^{i-1}$. It is easy to check that given two polynomials $a, p \in \mathcal{R}_q$ the product $a \cdot p \in \mathcal{R}_q$ is equivalent to the product of the matrix $\mathbf{A}$ with the vector $\mathbf{p} = (p_1, \dots, p_n)$. This allows us to perform polynomial multiplications in $\mathcal{O}(n\log(n))$ time complexity and $\mathcal{O}(\log(n))$ parallel depth if we use the Fast Fourier Transform.

Moreover, the quadratic space complexity issue is solved as we are able to represent an ideal lattice with only n values. Finally, security is guaranteed since reductions among some instances of the problems of section 3.2 are kept when considering them over ideal lattices [LPR13].

3.4 Ring learning with errors

Let n and q be integers, $\mathcal{R} = \mathbb{Z}[X]/\langle f(X)\rangle$ with $\deg(f) = n$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let χ_σ a discrete probability distribution over $\mathcal{R}$ (usually a Gaussian distribution) with standard deviation σ and a secret polynomial $s \in \mathcal{R}_q$.

Definition 3.17 (Ring Learning With Errors Distribution)**.** The RLWE distribution $\mathcal{L}_{s,\chi}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is sampled by choosing $a \xleftarrow{\mathcal{R}} \mathcal{R}_q$, $e \xleftarrow{\mathcal{R}} \chi_\sigma$ and outputting $(a, b = a \cdot s + e \mod q)$.

Definition 3.18 (Search-RLWE Problem)**.** Given m independent samples $(a_i, b_i) \xleftarrow{\mathcal{R}} \mathcal{L}_{\mathbf{s},\chi}$ for a fixed uniformly random s, find s.

Definition 3.19 (Decision-RLWE Problem)**.** Given m independent samples (a_i, b_i), decide whether this samples are distributed according to $\mathcal{L}_{s,\chi_\sigma}$ for a fixed uniformly random s; or according to an uniform distribution over $\mathcal{R}_q \times \mathcal{R}_q$.

Hardness of RLWE comes for large enough choices of q. Solving certain instantiations of Search-RLWE is as hard as quantumly solving a γ-SVP when $\gamma(n) \approx \text{poly}(n)$ on any ideal lattice.

The mechanism that we will use to encrypt the votes in an electronic voting is based on the RLWE problem. This mechanism is explained in section 4.1.1.

4. Public-key cryptography

4.1 Public-key encryption

Definition 4.1 (Public-key encryption scheme)**.** A public-key encryption scheme $\mathcal{E}$ is a triple of efficient algorithms $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec})$ that works as follows:

- **Gen:** The generator algorithm, takes as input the security parameter 1^λ and computes a public key and a secret key $(pk, sk) \xleftarrow{\mathcal{R}} \text{Gen}(1^\lambda)$ and finite sets $\mathcal{M}_\lambda$ and $\mathcal{C}_\lambda$.
- **Enc:** The encoding algorithm, takes a public key pk and the message $m \in \mathcal{M}_\lambda$ we wish to encrypt and computes a ciphertext $c \xleftarrow{\mathcal{R}} \text{Enc}(pk, m)$ belonging to $\mathcal{C}_\lambda$.
- **Dec:** The decoding algorithm, takes a secret key sk and a ciphertext $c \in \mathcal{C}_\lambda$ and computes $m \leftarrow \text{Dec}(sk, c)$, where m can be a message in $\mathcal{M}_\lambda$ or a special 'failure' value, clearly disinguishable from all normal messages. Note that, in contrast with Gen and Enc, the Dec algorithm is deterministic.

We also require the encryption scheme to work properly so decryption undoes encryption with overwhelming probability, that is, we require that this *correctness property* holds for every $(pk, sk) \xleftarrow{\mathcal{R}} \text{Gen}(1^\lambda)$ and every $m \in \mathcal{M}_\lambda$:

$$1 - \Pr\left[m = \text{Dec}(sk, \text{Enc}(pk, m))\right] \in \text{negl}(\lambda).$$

Intuitively, for an encryption scheme to be secure we need that, for any ciphertext c, it is impossible for any adversary $\mathcal{A}$ to recover the plaintext m unless it knows the secret key sk. This intuition leads to the notion of Public Key Encryption One-Way Chosen Plaintext Attack Security (PKE-OW-CPA).

Definition 4.2 (PKE-OW-CPA Security)**.** An encryption scheme $\mathcal{E} = (Gen, Enc, Dec)$ is said to be PKE-OW-CPA secure if for every PPT adversary $\mathcal{A}$ the following holds:

$$\Pr\left[m = \widehat{m} \,\middle|\, \begin{array}{c} (pk, sk) \xleftarrow{\mathcal{R}} \text{Gen}(1^\lambda);\ m \xleftarrow{\mathcal{R}} \mathcal{M}_\lambda; \\ c \xleftarrow{\mathcal{R}} \text{Enc}(pk, m);\ \widehat{m} \xleftarrow{\mathcal{R}} \mathcal{A}(1^\lambda, pk, c) \end{array}\right] \in \text{negl}(\lambda)$$

Although this seems like a strong definition of security, the reality is that information can be obtained from an encryption even if the message is not fully recovered. For instance, a scheme that encrypts only the first half of a message and leaves the second half unchanged can be OW-CPA secure, if the encrypted half is well encrypted, but the second half of the cipher might reveal crucial information. Thus, it is necessary to define a stronger notion of security. The

idea will be to encrypt data in such a way that it is indistinguishable from a random ciphertext. A possible approach to this idea is to design a scheme that, given two different messages and an encryption of one of them, it is hard to find out which one of the original messages has been encrypted. The definition of Public Key Encryption Indistinguishable Chosen Plaintext Attack Security (PKE-IND-CPA) arises from this approach

Definition 4.3 (PKE-IND-CPA Security)**.** An encryption scheme $\mathcal{E} = (Gen, Enc, Dec)$ is said to be PKE-IND-CPA secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following holds:

$$\left| \Pr \left[b = \widehat{b} \;\middle|\; \begin{array}{l} (pk, sk) \xleftarrow{\mathcal{R}} \mathsf{Gen}(1^\lambda);\ (m_0, m_1) \xleftarrow{\mathcal{R}} \mathcal{A}_1(1^\lambda, pk); \\ b \xleftarrow{\mathcal{R}} \{0,1\};\ c \xleftarrow{\mathcal{R}} \mathsf{Enc}(pk, m_b);\ \widehat{b} \xleftarrow{\mathcal{R}} \mathcal{A}_2(1^\lambda, c) \end{array} \right] - \frac{1}{2} \right| \in \mathsf{negl}(\lambda)$$

Although is not indicated, there can be communication between both parts of the adversary. That is, $\mathcal{A}_1$ might generate some information that $\mathcal{A}_2$ could take as an input.

IND-CPA security is also called semantic security.

Note that the demand on this definition of security is that the probability of correctly guessing the bit is at a negligible distance from $1/2$, which means that the adversary is practically outputting results randomly. Moreover, observe that this definition implies the necessity for the encryption to be probabilistic. If the encryption is deterministic, the adversary can simply encrypt one of the messages (for example m_0), compare it to the ciphertext and return $\widehat{b} = 0$ if they coincide and $\widehat{b} = 1$ otherwise. The adversary achieves a winning probability of 1.

There are several ways to improve the concept of security, for instance, providing the adversary more resources. From the definition of PKE-IND-CPA Security we may give to the adversary access to an oracle, in order to make the adversary able to decrypt its own ciphertexts (as long as they do not coincide with the given one) before trying to guess the bit. This notion of security is called Public Key Encryption Indistinguishable Chosen Ciphertext Attack Security (PKE-IND-CCA).

Definition 4.4 (PKE-IND-CCA Security)**.** An encryption scheme $\mathcal{E} = (Gen, Enc, Dec)$ is said to be PKE-IND-CCA secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following holds:

$$\left| \Pr \left[b = \widehat{b} \;\middle|\; \begin{array}{l} (pk, sk) \xleftarrow{\mathcal{R}} \mathsf{Gen}(1^\lambda);\ (m_0, m_1) \xleftarrow{\mathcal{R}} \mathcal{A}_1^{\mathcal{O}_{\mathrm{Dec}}}(1^\lambda, pk); \\ b \xleftarrow{\mathcal{R}} \{0,1\};\ c \xleftarrow{\mathcal{R}} \mathsf{Enc}(pk, m_b);\ \widehat{b} \xleftarrow{\mathcal{R}} \mathcal{A}_2^{\mathcal{O}_{\mathrm{Dec}}}(1^\lambda, c) \end{array} \right] - \frac{1}{2} \right| \in \mathsf{negl}(\lambda)$$

where the behaviour of the oracle is as follows:

$$\mathcal{O}_{\mathrm{Dec}}(c') = \begin{cases} \mathsf{Dec}(sk, c') & \text{if } c' \neq c \\ \text{'failure'} & \text{if } c' = c \end{cases}$$

IND-CCA security is also called active security.

This notion implies a high level of security on the scheme, since the decoding oracle makes the adversary really powerful. Nevertheless, reaching this level of security implies that the scheme must not have some homomorphic properties, which may be useful in some scenarios such as electronic voting.

4.1.1 RLWE encryption scheme

RLWE encryption scheme was first proposed by Lyubachevsky, Peikert and Regev in [LPR13], and it is based on the hardness of RLWE problems.

The encryption scheme works as follows:

Definition 4.5 (RLWE encryption scheme)**.** We consider the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ with n a power of 2 and q a prime. Messages are strings of n bits encoded as a polynomial in $\mathcal{R}_q$. The error distribution χ_σ will be the discrete Gaussian distribution with standard deviation $\sigma = \alpha q/\sqrt{2\pi}$ over $\mathcal{R}$, which will give 'small' elements of $\mathcal{R}_q$.

- **Gen**(1^λ): Compute suitable n and q according to λ. Choose $a \xleftarrow{\mathcal{R}} \mathcal{R}_q$ and small $s, e \xleftarrow{\mathcal{R}} \chi_\sigma$. Output $sk = s$ and $pk = (a, b = a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q$.
- **Enc**(pk, z, r, e_1, e_2): To encrypt a message $z \in \{0,1\}^n$ we view it as an element in $\mathcal{R}_q$ by using its bits as the 0-1 coefficients of a polynomial. Then we choose small elements $r, e_1, e_2 \xleftarrow{\mathcal{R}} \chi_\sigma$ and output $(u, v) = (r \cdot a + e_1, b \cdot r + e_2 + \lfloor \frac{q}{2} \rceil z) \in \mathcal{R}_q \times \mathcal{R}_q$
- **Dec**$(sk, (u, v))$: Compute:

$$\begin{aligned} v - u \cdot s &= b \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil z - (r \cdot a + e_1) \cdot s \\ &= (a \cdot s + e) \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil z - r \cdot a \cdot s - e_1 \cdot s \\ &= a \cdot s \cdot r + e \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil z - r \cdot a \cdot s - e_1 \cdot s \\ &= (e \cdot r - e_1 \cdot s + e_2) + \left\lfloor \frac{q}{2} \right\rceil z \\ &\approx \left\lfloor \frac{q}{2} \right\rceil z \end{aligned}$$

The choice of some parameters for RLWE and other lattice based cryptosystems is an open question, but there are some proposals in the literature that take into account different factors such as security levels or attacker types [RS10], requirements of security reductions [Reg05] or upper bounds on the decryption error probability [LP10].

The choices of q and σ determine if the encryption scheme works properly. If they are correct, the coefficients of $e \cdot r - e_1 \cdot s + e_2$ can be upper-bounded by less than $\frac{q}{4}$ and the

message z is recovered by rounding each coefficient of $v - u \cdot s$ to 0 or $\lfloor \frac{q}{2} \rceil$, whichever is closest modulo q. Also, this scheme allows to define a new algorithm **Re-Enc** to re-encrypt previously encrypted data. This algorithm works as follows:

- **Re-Enc**$(pk, (u, v), r', e_1', e_2')$: To re-encrypt a message z encrypted as (u, v) we choose small $r', e_1', e_2' \xleftarrow{\mathcal{R}} \chi_\sigma$ and output the pair

$$(u', v') = (u, v) + Enc(pk, 0, r', e_1', e_2') \in \mathcal{R}_q \times \mathcal{R}_q$$

This homomorphic operation preserves the plaintext z, although the error terms may grow linearly when applying re-encryption several times. To avoid potential mistakes, the number of re-encryptions must be taken into account when choosing q and σ, in order to achieve a sufficiently small error term even after applying as much re-encryptions as desired. Details about the error distribution and probabilities can be found in [San16].

This cryptosystem is used in many applications such as electronic voting, and it is the encryption scheme that we will consider for our purposes. It has been proved that the RLWE encryption scheme is IND-CPA secure, assuming the hardness of RLWE problems [LPR13].

4.2 Trapdoor functions

Encryption schemes admit some mathematical abstraction. In particular, they imply the existence of families of functions which are easy to apply (encryption) but difficult to invert (decryption) unless a secret key is known. Formally, given a family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{F}_\lambda = \{f_k : \mathcal{X}_k \to \mathcal{Y}_k\}_{k \in \mathcal{K}_\lambda}$ we can define the following properties:

- **Efficiently computable.** The family is efficiently computable if there exists a PPT algorithm Eval such that

$$\text{Eval}(k, x) = f_k(x)$$

- **One-way.** The family is one-way if it is efficiently computable but for all PPT algorithm $\mathcal{A}$:

$$\Pr\left[\mathcal{A}(1^\lambda, k, y) \in f_k^{-1}(y) \mid k \xleftarrow{\mathcal{R}} K_\lambda;\ x \xleftarrow{\mathcal{R}} \mathcal{X}_\lambda;\ y \leftarrow f_k(x)\right] \in \text{negl}(\lambda)$$

- **Collision resistant.** The family is collision resistant if for all PPT $\mathcal{A}$ we have:

$$\Pr\left[f_k(x) = f_k(x') \mid k \xleftarrow{\mathcal{R}} \mathcal{K}_\lambda;\ (x, x') \xleftarrow{\mathcal{R}} \mathcal{A}(1^\lambda, k)\right] \in \text{negl}(\lambda).$$

When a function $f : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ of this kind "compresses" its input, i.e. $|\mathcal{X}_\lambda| > |\mathcal{Y}_\lambda|$, it is often called a *hash function*.

4.2.1 Ajtai's One-Way function

The most used one-way function in a lattice setting is the one stated by Miklos Ajtai in [Ajt96], and it is based on the hardness of the SIS problem. Given a ring $\mathcal{R}$ (usually $\mathcal{R} = \mathbb{Z}_q$ or $\mathcal{R} = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$) and a fixed matrix $\mathbf{A} \in \mathcal{R}^{n\times m}$ this function is defined as:

$$\begin{array}{rccc} f_{\mathbf{A}}: & \mathcal{R}^m & \longrightarrow & \mathcal{R}^n \\ & \mathbf{s} & \longmapsto & \mathbf{As} \end{array}$$

Therefore, the SIS problem consists in finding small elements in the kernel of this application. Observe that for any $\mathbf{A}$, once discovered a solution $\mathbf{s}$ for this matrix, any extension $[\mathbf{A} \mid \mathbf{A'}]$ admits the vector $\mathbf{s'} = \begin{bmatrix} \mathbf{s} \\ \mathbf{0} \end{bmatrix}$ as a solution. Therefore, the SIS problem does not get harder as m increases. In fact, some columns $\mathbf{a_i}$ can be ignored as desired, making the problem easier.

For an instance of a SIS problem with $\mathcal{R} = \mathbb{Z}_q$, if γ and m are not large enough it could happen that there exists no solution. An easy pigeon-hole argument proves that when $\gamma \geq \sqrt{\lceil n \log q \rceil}$ and $m \geq \lceil n \log q \rceil$, the existence of a solution is guaranteed. Indeed, if this is the case, we can assume without loss of generality $m = \lceil n \log q \rceil$. Then there are more than q^m vectors $\mathbf{x} \in \{0,1\}^m$, and there must be two of them $\mathbf{x}$ and $\mathbf{x'}$ such that $\mathbf{Ax} = \mathbf{Ax'} \in \mathbb{Z}_q^n$, so $\mathbf{x} - \mathbf{x'} \in \{-1, 0, 1\}^m$ is a solution, since $\|\mathbf{x} - \mathbf{x'}\|_2 \leq \|\mathbf{1}\|_2 = \sqrt{m}$ and by hypothesis $\sqrt{m} \leq \gamma$.

The previous result implies that the Ajtai's function family $\{f_{\mathbf{A}} : \mathbb{Z}_q^m \to \mathbb{Z}_q^n\}$ is collision resistant, assuming SIS hardness, since a collision $(\mathbf{x}, \mathbf{x'})$ immediately yields a solution for SIS.

Micciano and Regev stated in [MR09] that for a large m, one should solve SIS for a submatrix where the number of columns is $\sqrt{n \log q / log \delta}$, for some constant δ, and expectation is to find a vector of length approximately $\min\{q, 2^{\sqrt{n \log q \log \delta}}\}$. The value δ is related to the optimal block-size in BKZ reduction [GN08] and in optimal lattice reductions is $\delta \approx 1.005$.

4.3 Commitment schemes

Suppose we want to send a message and keep it secret for a period of time. A commitment scheme allows us to hide the message until we want to reveal it and prove that it was not modified during the time it remained secret. Those two properties are called respectively hiding and binding.

Definition 4.6 (Non-interactive Commitment Scheme)**.** A non-interactive commitment scheme is a pair of efficient algorithms (Gen, Com) that works as follows:

- **Gen:** The generator algorithm, takes as input the security parameter 1^λ and computes a commitment key $ck \xleftarrow{\mathcal{R}} \text{Gen}(1^\lambda)$ and finite spaces $\mathcal{C}_{ck}$, $\mathcal{R}_{ck}$ and $\mathcal{M}_{ck}$. It also generates an efficiently sampleable probability distribution $D_{\mathcal{R}_{ck}}$ and a binding set $\mathcal{B}_{ck} \subset \mathcal{M}_{ck} \times \mathcal{R}_{ck}$.

- **Com:** The committing algorithm, takes the message $m \in \mathcal{M}_{ck}$ we wish to commit to and randomness $r \xleftarrow{\mathcal{R}} D_{\mathcal{R}_{ck}}$ and computes a commitment $c \leftarrow \mathsf{Com}_{ck}(m; r)$, where $c \in \mathcal{C}_{ck}$.

Usually, given a commitment c, we say that m is a *valid opening* for c if $c = \mathsf{Com}(m; r)$ for some randomness r (maybe with overwhelming probability). The binding property ensures that in such a case, m was the original message to which we had committed. It is necessary to dispose of a mechanism to verify if a given message is a valid opening for a given commitment. In fact, some commitment schemes include a third algorithm **Ver** that verifies openings of commitments.

As indicated above, the commitment scheme must satisfy the properties of hiding and binding. Specifically, we will require these properties to hold against PPT adversaries. In other words, adversaries will not be able neither to discover the committed value (computationally hiding) nor to find a commitment that can be opened to more than one message (computationally binding). Formally:

- **Computationally hiding.** A commitment scheme is said to be computationally hiding if for all interactive PPT adversaries $\mathcal{A}$:

$$\left| \Pr\left[\widehat{b} = b \;\middle|\; \begin{array}{c} ck \xleftarrow{\mathcal{R}} \mathsf{Gen}(1^\lambda);\ (m_0, m_1) \xleftarrow{\mathcal{R}} \mathcal{A}(ck) \\ b \xleftarrow{\mathcal{R}} \{0,1\};\ c \xleftarrow{\mathcal{R}} \mathsf{Com}_{ck}(m_b);\ \widehat{b} \xleftarrow{\mathcal{R}} \mathcal{A}(c) \end{array} \right] - \frac{1}{2} \right| \in \mathsf{negl}(\lambda)$$

- **Computationally binding.** A commitment scheme is said to be computationally binding if for all interactive PPT adversaries $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} m_0 \neq m_1 \wedge \\ \mathsf{Com}_{ck}(m_0; r_0) = \mathsf{Com}_{ck}(m_1; r_1) \end{array} \;\middle|\; \begin{array}{c} ck \xleftarrow{\mathcal{R}} \mathsf{Gen}(1^\lambda); \\ (m_0, r_0, m_1, r_1) \xleftarrow{\mathcal{R}} \mathcal{A}(ck) \end{array} \right] \in \mathsf{negl}(\lambda)$$

Definition 4.7 (Homomorphic commitment)**.** The commitment scheme is homomorphic if the message, randomness and commitment spaces are additive abelian groups and we have that for all $\lambda \in \mathbb{N}$, and for all $ck \xleftarrow{\mathcal{R}} \mathsf{Gen}(1^\lambda)$, for all $m_0, m_1 \in \mathcal{M}_{ck}$ and $r_0, r_1 \in \mathcal{R}_{ck}$:

$$\mathsf{Com}_{ck}(m_0; r_0) + \mathsf{Com}_{ck}(m_1; r_1) = \mathsf{Com}_{ck}(m_0 + m_1; r_0 + r_1)$$

Definition 4.8 (Compressing commitment)**.** A commitment scheme is said to be compressing if the size of the commitments are smaller than the size of the committed values.

4.3.1 A SIS-based commitment scheme

Baum *et al.* proposed in [BBC+18] a compressing commitment scheme based on the hardness of SIS. In particular it uses Ajtai's function

$$f_{\mathbf{A}}(\mathbf{s}) = \mathbf{A}\mathbf{s}$$

for a secret $\mathbf{s} \in \mathcal{R}^n$ and $\mathbf{A} \in \mathcal{R}^{r \times n}$. This scheme uses the fact that SIS hardness remains unaffected as n increases, as stated in section 4.2.1.

The scheme allows to commit messages in $\mathbb{Z}_q$ with $q < \bar{q}$, and uses uniformly random matrices $\mathbf{A_1} \in \mathbb{Z}_{\bar{q}}^{r \times 2r \log_q \bar{q}}$ and $\mathbf{A_2} \in \mathbb{Z}_{\bar{q}}^{r \times n}$ as the commitment key, where n is the number of elements we want to commit to. To commit n messages $\mathbf{m} \in \mathbb{Z}_q^n$, we pick $\mathbf{r} \xleftarrow{\mathcal{R}} \mathbb{Z}_q^{2r \log_q \bar{q}}$ and output the commitment vector $\mathbf{v} = \mathbf{A_1 r} + \mathbf{A_2 m} \mod \bar{q}$.

The whole commitment scheme is as follows:

- Gen(1^λ): Select a ring $\mathcal{R}$ ($\mathbb{Z}$ or $\mathbb{Z}[X]/\langle X^d + 1 \rangle$) and let $\mathcal{R}_{\bar{q}} = \mathcal{R}/\bar{q}\mathcal{R}$.
  Select parameters $q, \bar{q}, n, v, N, B, \sigma$.
  Pick uniformly random matrices $\mathbf{A_1} \xleftarrow{\mathcal{R}} \mathcal{R}_{\bar{q}}^{r \times r \log_q \bar{q}}$ and $\mathbf{A_2} \xleftarrow{\mathcal{R}} \mathcal{R}_{\bar{q}}^{r \times n}$.
  Return $ck = (q, \bar{q}, n, v, l, N, B, \mathcal{R}_{\bar{q}}, \mathbf{A_1}, \mathbf{A_2})$. The commitment key defines the spaces:

$$\mathcal{M}_{ck} = \mathcal{R}_{\bar{q}}^n \quad \mathcal{R}_{ck} = \mathcal{R}_{\bar{q}}^{2r \log_q \bar{q}} \quad \mathcal{C}_{ck} = \mathcal{R}_{\bar{q}}^r$$

$$\mathcal{B}_{ck} = \left\{ \mathbf{s} = \begin{bmatrix} \mathbf{m} \\ \mathbf{r} \end{bmatrix} \in \mathcal{R}_{\bar{q}}^{n+2r \log_q \bar{q}} \;\middle|\; \|\mathbf{s}\|_2 < B \right\} \quad D_{\mathcal{R}_{ck}} = D_\sigma^r$$

- $\mathrm{Com}_{ck}(\mathbf{m}; \mathbf{r})$: Return $\mathbf{c} = \mathbf{A_1 r} + \mathbf{A_2 s}$

It is also possible to commit to multiple vectors if we translate the scheme to a matrix notation. For instance, the commitment of l messages $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_l]$ with randomness $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_l]$ would be $\mathbf{C} = \mathrm{Com}_{ck}(\mathbf{M}; \mathbf{R}) = [\mathbf{c}_1, \ldots, \mathbf{c}_l]$.

The scheme is hiding due to the leftover hash lemma [HILL99] which shows that $(\mathbf{A_1 r}, \mathbf{A_1} \mod q)$ is statistically close to uniform. The binding property is a direct consequence of the hardness of the SIS problem. If there were $(\mathbf{r}, \mathbf{m}) \neq (\mathbf{r'}, \mathbf{m'})$ such that $\mathbf{A_1 r} + \mathbf{A_2 m} = \mathbf{A_1 m'} + \mathbf{A_2 r'}$, the vector $\mathbf{s} = \begin{bmatrix} \mathbf{r} - \mathbf{r'} \\ \mathbf{m} - \mathbf{m'} \end{bmatrix}$ is a solution for the SIS problem for the matrix $\mathbf{A} = \begin{bmatrix} \mathbf{A_1} & \mathbf{A_2} \end{bmatrix}$, if the parameters of the scheme are correctly chosen.

Notice that this commitment scheme would be compressing. Indeed, to commit a secret $\mathbf{s} \in \mathbb{Z}_q^n$ we take public random matrices $\mathbf{A_1} \in \mathbb{Z}_{\bar{q}}^{r \times 2r \log_q \bar{q}}$ and $\mathbf{A_2} \in \mathbb{Z}_{\bar{q}}^{r \times n}$ and a random vector $\mathbf{r} \in \mathbb{Z}_q^{2r \log_q \bar{q}}$. Hence we need $r \log \bar{q}$ bits to commit to n values over $\mathbb{Z}_q$. With a good choice of parameters (see table 2) the commitment scheme will still be computationally binding based on the worst-case hardness of SIVP.

4.4 Zero-Knowledge proofs of knowledge

Sometimes in cryptographic protocols is necessary to prove some knowledge, without revealing information at all. Zero-Knowledge proofs are mechanisms that allow us to attain this, so they

play a crucial part in applications such as authentication protocols, electronic voting systems, encryption primitives, multi-party computation schemes, and verifiable computation protocols.

To formalize the concept of zero-knowledge proof, let $\mathcal{L}$ be a language in NP and a binary relation $\mathcal{R} \subset \mathcal{L} \times \mathcal{L}$. We call *statement* to any $x \in \mathcal{L}$, and we say that ω is a *witness* for x if $(x, \omega) \in \mathcal{R}$. With this background, we can introduce the concept of Σ-Protocol.

Definition 4.9 (Σ-Protocol)**.** A Σ-Protocol is a protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ where, given a statement x, $\mathcal{P}$ tries to convince $\mathcal{V}$ that he knows a witness ω for x. We use the following notation:

$$ZKP\,[\,\omega \mid (x, \omega) \in \mathcal{R}]$$

Usually, this protocol has three steps:

1. **Commitment**. The prover $\mathcal{P}$ sends a commitment message t to the verifier $\mathcal{V}$
2. **Challenge**. The verifier sends a random challenge c to the prover.
3. **Final answer**. The prover sends an answer z and the verifies accepts or rejects the proof checking the conversation $tr = (x, t, c, z)$. If so, the tuple (x, t, c, z) is called an accepting conversation. Sometimes we consider that the verifier outputs a bit, with value 1 if he accepts the proof and 0 otherwise.

While this is the most common structure of a Σ-Protocol, with this 3-move setting that resembles the shape of the Σ, notice that is possible to perform more moves if needed. Nonetheless, the underlying behaviour of the protocol is always the same.

The prover may have to demonstrate to a large number of verifiers. Engaging in a proof with each one of them would suppose a great waste of time, considering that the verifier's task is simply to send challenges at random. It is possible to transform an interactive proof to a non-interactive proof without a specific verifier, where the prover gets the challenges by himself as the result of a hash function applied to the commitment. This procedure is called the Fiat-Shamir heuristic [FS87].

Also, we introduce these three properties:

- **Completeness.** If $\mathcal{P}$ is honest and does know a witness ω for x, then $\mathcal{V}$ always accepts the proof.
- **Soundness.** For any pair of accepting conversations $tr = (x, t, c, z)$, $tr' = (x, t, c', z')$ with $c \neq c'$ it is possible to extract a witness ω such that $(x, \omega) \in \mathcal{R}$.
- **Honest-Verifier Zero-Knowledge (HVZK).** The verifier learns nothing about the witness but the fact that the prover knows it.

Definition 4.10 (Zero-Knowledge proof of knowledge)**.** A zero-knowledge proof of knowledge is a Σ-Protocol that verifies completeness, soundness and HVZK.

There are several ways to mathematically formalize the properties stated above. In particular we are interested in the ones used for the arguments in [BCC+16, BBC+18] which will be the arguments we will adapt to our purposes. These formalized concepts consider some public information called common reference string known by all parties in the protocol.

- **Statistical Completeness.** Given a function $\rho : \mathbb{N} \to [0,1]$, then the protocol has statistical completeness with completeness error ρ if for all adversaries $\mathcal{A}$:

$$\Pr\left[(x,\omega) \in \mathcal{R} \ \wedge \ b = 0 \ \middle| \ \begin{array}{c} (x,\omega) \xleftarrow{\mathcal{R}} \mathcal{A}; \\ (tr,b) \xleftarrow{\mathcal{R}} (\mathcal{P}(x,\omega),\mathcal{V}(x)) \end{array}\right] \leq \rho(\lambda)$$

- **Computational knowledge soundness.** Given a function $\epsilon : \mathbb{N} \to [0,1]$, then the protocol has computational knowledge soundness if for all PPT $\mathcal{P}^*$ there is a polynomial time extractor $\mathcal{E}$ such that for all PPT adversaries $\mathcal{A}$:

$$\Pr\left[(x,\omega) \notin \mathcal{R} \ \wedge \ b = 1 \ \middle| \ \begin{array}{c} (x,s) \xleftarrow{\mathcal{R}} \mathcal{A}\,(tr,b) \xleftarrow{\mathcal{R}} (\mathcal{P}^*(x,s),\mathcal{V}(x)) \\ \omega \xleftarrow{\mathcal{R}} \mathcal{E}^{\mathcal{P}^*(u,s)}(u,tr,b) \end{array}\right] \leq \epsilon(\lambda)$$

- **Special Honest-Verifier Zero-Knowledge (SHVZK)** A protocol is SHVZK if there exists a PPT simulator $\mathcal{S}$ such that for all interactive adversaries cA:

$$\begin{aligned} & \Pr\left[(x,\omega) \in \mathcal{R} \ \wedge \ \mathcal{A}(tr) = 1 \ \middle| \ (x,\omega,\varrho) \xleftarrow{\mathcal{R}} \mathcal{A};\ (tr,b) \xleftarrow{\mathcal{R}} (\mathcal{P}(x,\omega),\mathcal{V}(x;\varrho))\right] \\ \approx \ & \Pr\left[(x,\omega) \in \mathcal{R} \ \wedge \ \mathcal{A}(tr) = 1 \ \middle| \ (x,\omega,\varrho) \xleftarrow{\mathcal{R}} \mathcal{A};\ (tr,b) \xleftarrow{\mathcal{R}} \mathcal{S}(x,\varrho)\right] \end{aligned}$$

  where ϱ is the randomness of the verifier.

It is obvious that the completeness property is necessary so the protocol works in a desirable way. The objective of the soundness property is to bound the probability that a dishonest prover can fool the verifier. Consider a (maybe dishonest) prover. Once the commitment is sent, if the prover is indeed dishonest he will only be able to answer one of the possible challenges and fool the verifier. Since the challenge set size is usually exponential, the probability of this to happen is negligible. If he were able to answer to two different challenges correctly, he would be able to efficiently extract a witness, and therefore he would be honest. Finally, honest-verifier zero-knowledge implies that the verifier has not learned any new information from the prover, since it is possible to simulate the proof without knowing a witness if the challenges are known in advance.

Definition 4.11 (Zero-Knowledge argument of knowledge)**.** A Zero-Knowledge argument of knowledge is a Σ-Protocol that verifies statistical completeness, computational knowledge soundness and SHVZK.

4.4.1 Zero-knowledge argument for preimages of Ajtai's function

Here we present a zero-knowledge argument for l equations given by Ajtai's one-way function over a ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ (with $\mathcal{R} = \mathbb{Z}$ or $\mathcal{R} = \mathbb{Z}[X]/\langle X^d + 1\rangle$), proposed in [BBC+18]. Given matrices $\mathbf{A} \in \mathcal{R}_q^{r\times v}$ and $\mathbf{T} \in \mathcal{R}_q^{r\times l}$, with this method the prover can prove knowledge of a witness $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_l] \in \mathcal{R}^{v\times l}$ with $\|\mathbf{s}_i\|_2 \leq \beta$ such that $\mathbf{AS} = c \cdot \mathbf{T}$. In particular, this proof allows to prove knowledge of committed messages using the commitment scheme 4.3.1, and therefore proving that those messages were fixed in advance.

$$ZKP\left[\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_l] \in \mathcal{R}^{v\times l} \;\middle|\; \begin{array}{l}\mathbf{AS} = c\mathbf{T} \\ \|\mathbf{s}_i\|_2 \leq \beta\end{array}\right]$$

Protocol 1: Proof for Ajtai's function preimages

$\mathcal{P}(\mathbf{A}, \mathbf{S}, \mathbf{T})$		$\mathcal{V}(\mathbf{A}, \mathbf{T})$
$\mathbf{Y} \xleftarrow{\mathcal{R}} D_\sigma^{v\times n}$ $\mathbf{W} = \mathbf{AY}$		
	$\xrightarrow{\mathbf{W}}$	
		$\mathbf{C} \xleftarrow{\mathcal{R}} \mathcal{C}^{l\times n}$
	$\xleftarrow{\mathbf{C}}$	
$\mathbf{Z} = \mathbf{SC} + \mathbf{Y}$ Abort if $\mathrm{Rej}(\mathbf{Z}, \mathbf{B}, \sigma, \rho) = 1$		
	$\xrightarrow{\mathbf{Z}}$	
		$\forall i \in [n] : \|z_i\|_2 \overset{?}{\leq} B$ $\mathbf{AZ} \overset{?}{=} \mathbf{TC} + \mathbf{W}$

The challenge set depends on the ring we are working on. For $\mathcal{R} = \mathbb{Z}$, $\mathcal{C} = \{0, 1\}$, while for $\mathcal{R} = \mathbb{Z}[X]/\langle X^d + 1\rangle$, $\mathcal{C} = \{0\} \bigcup \{\pm X^j\}_{j<d}$

This argument does not exactly follow the definition given in section 4.4. First, we are slacking the condition of the preimages with the factor c. The proof over $\mathbb{Z}$ achieves $c = 1$ and therefore an exact value, but for $\mathbb{Z}[X]/\langle X^d + 1\rangle$ it only guarantees $c = 2$. Despite this 'slack', it is still good enough for many applications since it allows to prove knowledge of the plaintext. In fact, over the polynomial ring $\mathbb{Z}[X]/\langle X^d + 1\rangle$ the proof can be shorter, since we can pick a challenge matrix with a number of columns approximately $\lambda/\log 2d$, due to the invertibility of the difference of monomials in $\mathcal{R}$, stated by Benhamouda *et al.* in [BCK+14].

Secondly, since we are trying to prove knowledge of a matrix $\mathbf{S}$ in zero-knowledge, the output matrix $\mathbf{Z}$ must have an independent distribution from $\mathbf{S}$. During the protocol, the prover calculates $\mathbf{B}+\mathbf{Y}$, where $\mathbf{B}$ depends on the secret $\mathbf{S}$, and $\mathbf{Y}$ is sampled from a Gaussian distribution in order to 'mask´ this dependency. Therefore, to remove the dependency of $\mathbf{B}+\mathbf{Y}$ on $\mathbf{B}$ we must apply a technique that aborts the protocol if this dependency is noticeable. We can mathematically ensure the correct performance of this technique thanks to lemma 4.12, proposed by Vadim Lyubashevsky in [Lyu12], which uses the rejection sampling Rej (1). With this result the protocol would need in average ρ iterations until the challenge overpass the abort step, where ρ is a constant that depends on σ. Nevertheless, in practice we will be more interested in the non-interactive protocol obtained by Fiat-Shamir, which only outputs a proof after obtaining a challenge that passes the abort step.

Algorithm 1: Rej $(\mathbf{Z}, \mathbf{B}, \sigma, \rho)$

$u \xleftarrow{\mathcal{R}} [0,1)$;
if $u > \frac{1}{\rho} \cdot \exp\left(\frac{-2\langle \mathbf{Z},\mathbf{B}\rangle + \|\mathbf{B}\|_2^2}{2\sigma^2}\right)$ **then**
  return 0
else
  return 1

Lemma 4.12 ([Lyu12]). *Let $\mathbf{B} \in \mathcal{R}^{r\times n}$. Consider a procedure that samples $\mathbf{Y} \xleftarrow{\mathcal{R}} D_\sigma^{r\times n}$ and returns the output of* Rej$(\mathbf{Z} = \mathbf{Y}+\mathbf{B}, \mathbf{B}, \sigma, \rho)$ *with $\sigma \geq \frac{12}{\ln \rho} \cdot \|\mathbf{B}\|_2$. The probability for this procedure to output 1 is within 2^{-100} of $1/\rho$. The distribution of $\mathbf{Z}$, given the output is 1, is within statistical distance 2^{-100} of $D_\sigma^{r\times n}$.*

Finally we can state the main result from this section, and give the conditions that the parameters must fulfill to achieve a zero-knowledge proof of knowledge, for the case $\mathcal{R} = \mathbb{Z}$:

Theorem 4.13 ([BBC+18]). *Let $v, r \in poly(\lambda)$ and $n \geq \lambda + 2$. Let $s, \rho, \sigma \in \mathbb{R}$ such that s is an upper bound for $s_1(\mathbf{S})$, $\rho > 1$ a constant, $\sigma \geq \frac{12}{\ln \rho} s\sqrt{nl}$, and $B = \sqrt{2v}\sigma$. Then protocol described in section 4.4.1 is a zero-knowledge proof of knowledge.*

And for the case $\mathcal{R} = \mathbb{Z}/\langle X^d + 1\rangle$:

Theorem 4.14 ([BBC+18]). *Let $v, r \in poly(\lambda)$ and $n \geq \frac{\lambda+2}{\log(2d+1)}$. Let $s, \rho, \sigma \in \mathbb{R}$ such that s is an upper bound for $s_1(\mathbf{S})$, $\rho > 1$ a constant, $\sigma \geq \frac{12}{\ln \rho} s\sqrt{nl}$, and $B = \sqrt{2md}\sigma$. Then protocol described in section 4.4.1 is a zero-knowledge proof of knowledge.*

5. Arithmetic circuit satisfiability

5.1 Introduction to arithmetic circuits

An arithmetic circuit over a ring $\mathcal{R}$ is a directed acyclic graph whose vertices are called gates and edges are called wires. Gates of in-degree 0 are called input gates and usually are associated to variables or constants. The remaining gates are either multiplication gates or addition gates, labelled respectively with $\bigotimes$ and $\bigoplus$. We can consider only binary operations, therefore multiplication and addition gates have in-degree equal to 2, and out-degree equal to 1, although we allow the result to be attached to several other gates as input. In this case, we say the gate is fan-in 2 and operands are usually called *left* and *right*.

If we consider an arithmetic circuit with N fan-in 2 gates as described above there are in total $3N$ input and output wires feeding in and out the gates. An instance of an arithmetic circuit is a description of it, and includes a set of gates, the connection wires between them and known values assigned to some of the inputs and outputs. We say such circuit is *satisfiable* if there exists an assignment complying all the gates, the wires and the known values in the instance. More precisely, a witness for a given instance consists of assignments to the input and output wires of each gate such that:

- The known values match the corresponding assignments of the witness.
- The output of each gate corresponds to the result of applying the operation to the input wires of the gate
- The value of an output wire (or an input gate) matches the value of all input wires connected to it.

The natural behaviour of arithmetic circuits is calculating linear relations and checking constraints satisfiability. Even though the computational nature of an arithmetic circuit implies the impossibility of storing the symbolic expression of a polynomial, sometimes is desirable to prove equality between two polynomials. To sort out this issue we will use the Schwartz-Zippel lemma, which allows us to bound the probability of two polynomials being distinct when the evaluations are equal, if some conditions are met.

Lemma 5.1 (Schwartz-Zippel)**.** *Let p be a non-zero multivariate polynomial in $\mathbb{Z}_q[X_1, \dots, X_n]$ of degree d. Then the probability that $p(x_1, \dots, x_n) = 0$ for uniformly random chosen $x_1, \dots, x_n \xleftarrow{\mathcal{R}} \mathbb{Z}_q^*$ is at most $\frac{d}{q-1}$*

Informally, the Schwartz-Zippel lemma states that if we evaluate two polynomials several times on different random points and we always obtain the same value, the chances are that they are the same polynomial.

To construct arguments for satisfiabilty of the circuit we will explain the method proposed by Baum *et al.* [BBC+18]. The main idea is to use previous arguments of circuit satisfiability [BCC+16, Gro09] based on the discrete logarithm hardness assumption and adapt them to a lattice setting, in particular to the SIS assumption.

5.2 Reduction to polynomial equations

The argument for the satisfiability of an arithmetic circuit over $\mathbb{Z}_q$ can be reduced to the verification of multiplication constraints (arising from multiplication gates) and linear constraints (arising from addition gates and multiplication by constants). Then, it is possible to reduce again this set of constraints to two polynomial equations.

Suppose the arithmetic circuit consists in $N = nm$ multiplication gates. We can assume that the circuit has been pre-processed so the input and output wires feed into and go out from only multiplication gates (see [BCC+16] for details). Now, we define three $m \times n$ matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ such that their entries correspond to the left, right and output of each gate, respectively. To prove satisfiability of the arithmetic circuit, it must hold:

$$\mathbf{A} \circ \mathbf{B} = \mathbf{C} \tag{1}$$

where $\circ$ is the entry-wise product (also called Hadamard product), which yields a system of N equations.

Also, to keep track of the relations between outputs and inputs of the gates, we can express these constraints with $U < 2N$ equations of the form:

$$\sum_{i=1}^{m} \mathbf{a_i} \cdot \mathbf{w}_{u,a,i} + \sum_{i=1}^{m} \mathbf{b_i} \cdot \mathbf{w}_{u,b,i} + \sum_{i=1}^{m} \mathbf{c_i} \cdot \mathbf{w}_{u,c,i} = K_u \tag{2}$$

where $\mathbf{a_i}$, $\mathbf{b_i}$ and $\mathbf{c_i}$ are the rows of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. For instance, to express the relation

$$a_{1,1} + a_{1,2} = b_{1,1}$$

we will set:

$$\begin{aligned} U &= 1 \\ \mathbf{w}_{1,a,i} &= (1, 1, 0, \dots 0) \\ \mathbf{w}_{1,b,i} &= (-1, 0, \dots 0) \\ \mathbf{w}_{1,c,1} &= (0, \dots 0) \end{aligned}$$

and all the remaining vectors to zero.

With this method we have reduced the arithmetic circuit satisfiability to the verification of a set of $N + U$ equations. The next step is to translate those $N + U$ equations to two polynomial equations over a formal indeterminate Y.

For the N equations 1, if we consider the polynomials with i-th coefficients the terms of each side of the equations, we have that the N equations hold if and only if both polynomials are equal:

$$\sum_{i=1}^{m} \mathbf{a_i} \circ \mathbf{b_i} Y^i = \sum_{i=1}^{m} \mathbf{c_i} Y^i$$

We can also consider an analogous process for the U constraint equations on the wires, so they hold if and only if the polynomials are the same:

$$\sum_{u=1}^{U} \left(\sum_{i=1}^{m} \mathbf{a_i} \cdot \mathbf{w}_{u,a,i} + \sum_{i=1}^{m} \mathbf{b_i} \cdot \mathbf{w}_{u,b,i} + \sum_{i=1}^{m} \mathbf{c_i} \cdot \mathbf{w}_{u,c,i} \right) Y^u = \sum_{u=1}^{U} K_u Y^u$$

If we define the polynomials:

$$\mathbf{w}_{a,i}(Y) = \sum_{u=1}^{U} \mathbf{w}_{u,a,i} Y^{N+1+u}$$

$$\mathbf{w}_{b,i}(Y) = \sum_{u=1}^{U} \mathbf{w}_{u,b,i} Y^{N+1+u}$$

$$\mathbf{w}_{c,i}(Y) = \sum_{u=1}^{U} \mathbf{w}_{u,c,i} Y^{N+1+u}$$

$$K(Y) = \sum_{u=1}^{U} K_u Y^{N+1+u}$$

then the circuit is satisfied if and only if the following two equations hold:

$$\sum_{i=1}^{m} \mathbf{a_i} \cdot \mathbf{w}_{a,i}(Y) + \sum_{i=1}^{m} \mathbf{b_i} \cdot \mathbf{w}_{b,i}(Y) + \sum_{i=1}^{m} \mathbf{c_i} \cdot \mathbf{w}_{c,i}(Y) - K(Y) = 0 \tag{3}$$

$$\sum_{i=1}^{m} \mathbf{a_i} \circ \mathbf{b_i} Y^i = \sum_{i=1}^{m} \mathbf{c_i} Y^i \tag{4}$$

Now, to prove these equations hold, one can define polynomials which will have some terms equal to zero if the equations hold. This can be proved if the prover shows evaluations of these polynomials at random points to the verifier. For example, if we define

$$\mathbf{a}(X) = \mathbf{a}_0 + \sum_{i=1}^{m} \mathbf{a}_i y^i X^i$$

$$\mathbf{b}(X) = \mathbf{b}_{m+1} + \sum_{i=1}^{m} \mathbf{b}_i X^{m+1-i}$$

$$\mathbf{c} = \sum_{i=1}^{m} \mathbf{c}_i y^i$$

Then:

$$\mathbf{a}(X) \circ \mathbf{b}(X) \stackrel{?}{=} \mathbf{c}X^{m+1} + \sum_{\substack{i=0 \\ i \neq m+1}}^{2m} \mathbf{h}_i X^i$$

With this construction the equation 4 is verified if and only if the X^{m+1} term of $\mathbf{a}(X) \circ \mathbf{b}(X)$ is $\mathbf{c}$ and therefore we can check whether equation 4 is satisfied. An analogous way allows to prove if equation 3 is also satisfied.

Now, to apply the Schwartz-Zippel lemma, we require the base field to have a super-polynomial size over the security parameter λ. This is not the case, as this proof is considered for a smaller q, namely a polynomial over λ. To apply the lemma, following the argument of Cramer *et al.* [CDK14], the reduction to polynomial equations is extended to a field extension $GF(q^{2k}) = \mathbb{Z}_p[\phi]/\langle f(\phi) \rangle$. Therefore the size is exponentiated by a factor of $2k$ and the Schwartz-Zippel lemma ensures overwhelming soundness.

By choosing a good basis $\mathcal{B}$ for $GF(q^{2k})$ it is possible to perform multiplications of elements in this extension field without any overflow modulo f, and therefore simulate arithmetic on the extension using arithmetic in $\mathbb{Z}_q^{2k}$. In this case, if we want to multiply on the left by an element $\mathbf{x} \in GF(q^{2k})$, we can consider the matrix $\mathbf{M}_\mathbf{x}$ that simulates multiplication by $\mathbf{x}$ in $\mathbb{Z}_q^{2k}$. Now the procedure is analogous to the one over the smaller field, but considering $N = mnk$ multiplication gates and elements in $\mathbb{Z}_q^{2k \times n}$ as elements whose n columns are elements of $GF(q^{2k})$ (we refer to [BBC+18] for the details).

5.3 Arithmetic circuit satisfiability argument

Since we have reduced the proof of satisfiability to two polynomial equations, we will give two sub-protocols to verify each of these equations. Then we proceed to show the parameters which must be chosen carefully for our purposes. And finally we calculate the efficiency of the whole argument, and state that it has sub-linear communication.

It is important to notice that in a lattice setting we require the messages (in this case, the values of the wires that work as witnesses) to have small norm, in order to take advantage of the SIS assumption. In this case, for an arithmetic circuit which works on a field modulo p the aim is to preserve secrecy for the values of the wires, so we need to commit to these values over another prime which is greater than q, to make the openings small on this larger ring. In other words, although we are operating on $\mathbb{Z}_q$, we are considering the values on a larger $\mathbb{Z}_{\bar{q}}$.

The argument of section 4.4.1 provides a manner to check that preimages of Ajtai's function have small norm modulo a prime $\bar{q}$, which we can specify with the desired size. This may cause difficulties since values over the large ring can be different but congruent modulo q. In the final part of the proof, the prover will send an additional commitment $\mathbf{E}$ that contains a multiple of q to solve the issue.

5.3.1 Argument for multiplicative relations and linear constraints

Here we discuss the argument that allows a prover to prove knowledge of $N = nkm$ triples satisfying multiplicative relations. In it we will use an extension of the commitment scheme explained in section 4.3.1. Let $A \in \mathbb{Z}_{\bar{q}}^{2k \times n}$ and $R \in \mathbb{Z}_{\bar{q}}^{2k \times n'}$ and the extended commitment scheme Com_{ck}^* as

$$\mathrm{Com}_{ck}^*(A; R) = \begin{pmatrix} \mathrm{Com}_{ck}(\mathbf{a}_1; \mathbf{r}_1) \\ \mathrm{Com}_{ck}(\mathbf{a}_2; \mathbf{r}_2) \\ \vdots \\ \mathrm{Com}_{ck}(\mathbf{a}_{2k}; \mathbf{r}_{2k}) \end{pmatrix}$$

where $\mathbf{a}_i$ and $\mathbf{r}_i$ are the rows of A and R.

Both the prover and the verifier know the commitment key ck and the basis $\mathcal{B}$ which specifies how the multiplications are performed. The complete protocol

$$ZKP\left[A_i, B_i, C_i \in \mathbb{Z}_q^{k \times n} \;\middle|\; \forall i \in \{1, \dots, m\} : A_i \circ B_i \equiv C_i \mod q \right]$$

can be found in appendix A.

Secondly, to prove knowledge of linear constraints that are part of the arithmetic circuit alongside the product relations, we must give an argument to prove knowledge of vectors $\mathbf{a}_{i,j}$, $\mathbf{b}_{i,j}$, $\mathbf{c}_{i,j} \in \mathbb{Z}_q^n$, such that for a statement formed by vectors $\mathbf{w}_{u,a,i,j}$, $\mathbf{w}_{u,b,i,j}$, $\mathbf{w}_{u,c,i,j}$ arising from an arithmetic circuit instance, this equation hold:

$$\sum_{i=1,j=1}^{m,k} \mathbf{a_{i,j}} \cdot \mathbf{w}_{u,a,i,j} + \sum_{i=1,j=1}^{m,k} \mathbf{b_{i,j}} \cdot \mathbf{w}_{u,b,i,j} + \sum_{i=1,j=1}^{m,k} \mathbf{c_{i,j}} \cdot \mathbf{w}_{u,c,i,j} = K_u \qquad \forall u \in \{1, \dots, U\}$$

The complete protocol

$$ZKP\left[\mathbf{a}_{i,j},\ \mathbf{b}_{i,j},\ \mathbf{c}_{i,j} \in \mathbb{Z}_q^n \;\middle|\; \sum_{i=1,j=1}^{m,k} \mathbf{a_{i,j}} \cdot \mathbf{w}_{u,a,i,j} + \sum_{i=1,j=1}^{m,k} \mathbf{b_{i,j}} \cdot \mathbf{w}_{u,b,i,j} + \sum_{i=1,j=1}^{m,k} \mathbf{c_{i,j}} \cdot \mathbf{w}_{u,c,i,j} = K_u \right]$$

can be found in appendix B.

Combining both protocols, we are able to give a zero-knowledge proof for arithmetic circuit satisfiability. The prover commits to n wire values forming $\mathcal{O}(mk)$ commitments and runs both protocols, reusing the same commitments. If the verifier accepts both protocols, the prover will have proved knowledge of values that satisfy the arithmetic circuit.

5.3.2 Efficiency

Both arguments use 9 moves between prover and verifier (including the sub-protocols to prove knowledge for the commitments). Sub-linear communication is achieved if the choice of parameters is wise. Concretely, it is possible to achieve communication in $\mathcal{O}(\sqrt{N\lambda \log^3(N)})$ elements of $\mathbb{Z}_{\bar{q}}$, prover calculations in $\mathcal{O}(N \log(N)(\log \lambda)^2)$ bit operations and verifier calculations in $\mathcal{O}(N(\log \lambda)^3)$ bit operations. Moreover, a good choice of parameters is essential to prove that the argument fulfills the requirements to be a zero-knowledge proof. The proposal of Baum *et al.* [BBC+18] is to use the parameters according to table 2.

Parameter	Size	Description
λ		Security parameter
q	$\text{poly}(\lambda)$	Field for arithmetic circuit
N	$N = kmn \in \text{poly}(\lambda)$	Number of multiplication gates in the circuit
n	$n \approx \sqrt{\frac{Nr \log \bar{q}}{\lambda \log N\lambda q}}$	Length of vectors
k	$k \approx \lambda / \log_2 q$	Control of soundness error
m	$m = N/kn$	Number of commitments
$\bar{q}$	$\bar{q} \in \mathcal{O}(N^3k^2m^2q^4\sqrt{r})$	Modulus for SIS instances
r	$r \in \mathcal{O}(\log n)$	Size of the commitments

Table 2: Parameter selection.

5.4 Example: RLWE encryption proof

The aim of this section is to illustrate how arithmetic circuit satisfiability can be useful to model a real zero-knowledge proof in a simple cryptographic application. Since we will work with the RLWE encryption scheme, we will give an example of a proof of knowledge that will allow a prover $\mathcal{P}$ to implement a RLWE encryption of N messages and demonstrate that he performed it in an honest way. In other words, the prover will create a list $\{z_1, \dots, z_N\}$ of messages and will output a list $\{c_1, \dots, c_N\}$ of ciphertexts that will be public. The prover will prove knowledge of the random values $r^{(i)}, e_1^{(i)}, e_2^{(i)}$ for $i \in \{1, \dots, N\}$ without revealing them.

$$ZKP\left[z_i, r^{(i)}, e_1^{(i)}, e_2^{(i)}, \quad i \in [N] \;\middle|\; \forall i \in [N] : \left\{ \begin{array}{l} c_i = (u_i, v_i) = \mathrm{Enc}(pk, z_i, r^{(i)}, e_1^{(i)}, e_2^{(i)}) \\ \left\|r^{(i)}\right\|_2, \left\|e_1^{(i)}\right\|_2, \left\|e_2^{(i)}\right\|_2 \leq \delta \end{array} \right. \right]$$

To set up this proof we must craft an arithmetic circuit that executes the RLWE encryption. In this arithmetic circuit some values will be public and some other will remain private. The public ones will be part of the statement, so if $\mathcal{P}$ proves knowledge of the private values, it will be equivalent to prove knowledge of the random parameters of the encryption.

The arithmetic circuit of one encryption is given in figure 1, where the public values are colored in blue and the secret ones in red. To complete the whole circuit, just assemble N copies of this sub-circuit.

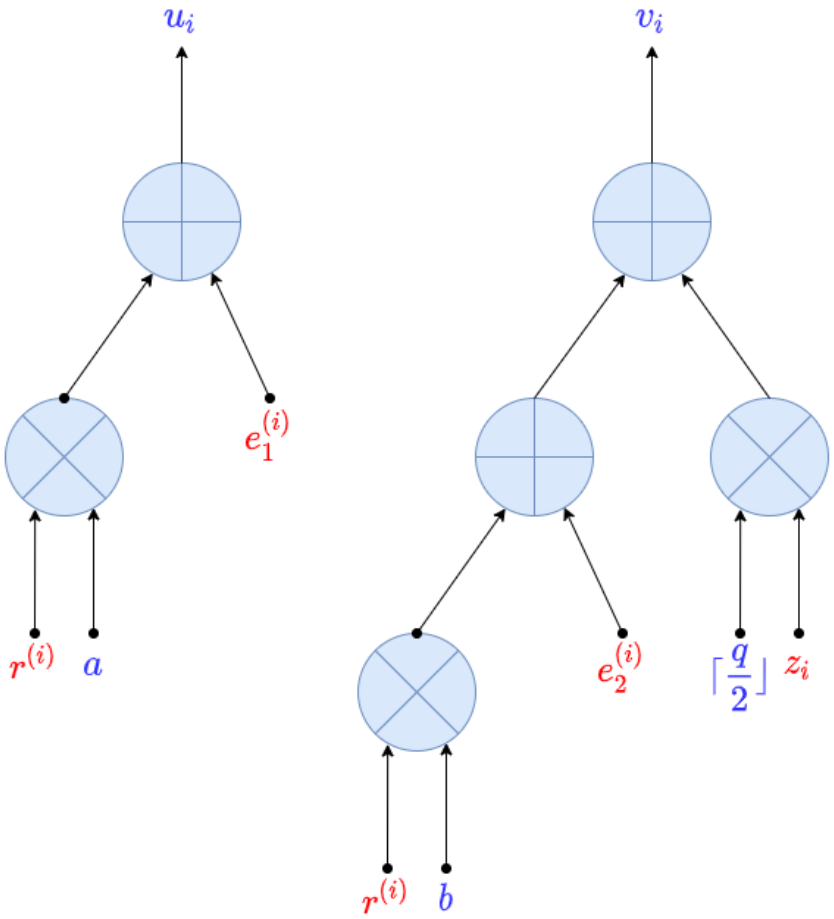

Figure 1: Arithmetic circuit for a RLWE encryption

To prove that the parameters of the encryption are small, we engage in a ZKP as explained in section 4.4.1 within the ambient field. The value of δ will depend on the total number of

re-encryptions we wish to perform after the first encryption, since the errors can stack and grow linearly in this amount of re-encryptions. If we wish to perform just one encryption, the value must verify that

$$\|e \cdot r - e_1 \cdot s + e_2\|_\infty \leq \frac{q}{4}$$

6. Proof of a shuffle

In this section we will explain the main result of this thesis. The goal is to define a shuffle in an electronic voting process and give a protocol in a lattice based setting to prove its correctness, using the results from the previous sections.

6.1 Voting process

In an electronic voting process the first step is the voter to cast its vote. At that time, the vote is encrypted and signed, and the signature is checked to be valid. Therefore every vote is unique and linked to the voter, so it is necessary to ensure that it is not possible to track a vote so that the option chosen by a voter can be found out. After the verification of the signature in the census, the encrypted vote (without the signature), goes forward to a shuffling phase. A shuffle is a mechanism whose purpose is to hide the origin of the votes in order to make this tracking difficult. Let us suppose that a total of N votes get to this phase. The shuffle phase consists in several mixing nodes that take the N ciphertexts as an input and then re-encrypt and randomly permute them, and output the result. The desired behaviour is that the input list and the output list contain the same information, i.e. the same plaintexts, but in a random order. This re-encrypting and permuting operation is called a shuffle. The set of mixing nodes is called a mixing net (mix-net) [Cha81], and it is commonly used in applications that demand privacy.

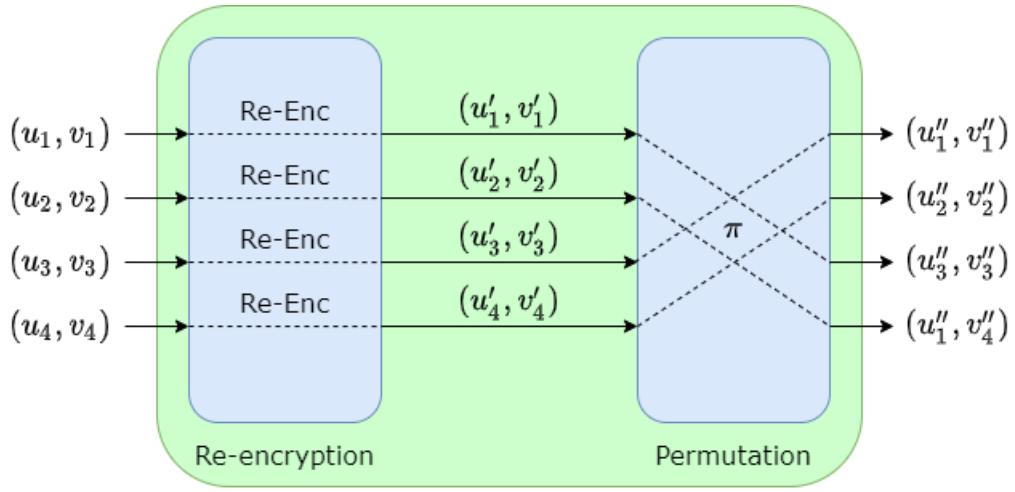

Figure 2: Mix node

Our objective is to arm each mixing node with a proof to verify that the performed shuffle is correct. Without this proof, a malicious node could replace some ciphertexts with another ones of his choice (for example, encrypted votes for a certain candidate) without being noticed

and therefore manipulating the results. This proof will allow to prove that the shuffle is correct, but it does not guarantee anonymity, since the prover can perform the shuffle with parameters not coming from the desired distributions and allow adversaries to trace back the input from the output. Despite this, the process is still anonymous if just one mixing node is honest and shuffles the list accordingly to the encrypting algorithm. Additionally, the mix-node must choose the re-encrypting parameters and the permutation at random and without revealing them, to preserve privacy of the votes. This is the expected behaviour. In standard voting, for our vote to remain anonymous we also need that at least one step of the process is honest. To deposit the vote inside an envelope is equivalent to encrypt the vote, and the ballot box acts as the random permutation.

The aim of this section is to reduce the proof of a shuffle to a zero-knowledge argument of satisfiability of a certain arithmetic circuit. Each node receives a list of N RLWE ciphertexts $C = \{c_1, \dots, c_N\}$ and outputs another list $D = \{d_1, \dots, d_N\}$. If we manage to describe an arithmetic circuit with the least possible amount of gates (i.e. an $\mathcal{O}(N \log(N))$ or less number of gates) such that the list D is the result of shuffling (re-randomizing and permuting) the list C, the communication efficiency for satisfiability stated in section 5.3.2 permits us to set a zero-knowledge argument for a shuffle in sub-linear communication time.

As seen in section 5.4, with some slight changes, we can construct an arithmetic circuit to perform a re-encryption (see figure 3).

Again, the public values are painted in blue and the private ones in red. Observe that we just need 6 gates to perform the re-encryption. This yields a sub-circuit of size $\mathcal{O}(N)$ gates to re-encrypt the N ciphertexts.

It remains to model the random permutation. In the first moment, the way we intended to solve it was to adapt the proof of a shuffle of Bayer and Groth in the discrete logarithm setting [BG12] to the lattice setting, in a similar way as [CMM17, Mar18], but using arithmetic circuits. This method consists in creating polynomials $F_{C'}$ and F_D in $\mathcal{R}_q[\Gamma]$ whose roots correspond to the elements of each list C' (obtained after the re-encryption of C) and D, respectively. If both lists have the same elements but in a different order, the polynomials must be equal. Using a generalized version of the Schwartz-Zippel lemma for general rings, and evaluating these polynomials in points chosen from a suitable subset, we can check whether they are equal or not. Nevertheless, over the ring $\mathcal{R}_q$ polynomials may have more roots than its degree. Therefore, the fact $F_{C'} = F_D$ does not imply $C' = D$, since a malicious prover could replace one element of the list C' with a root of F_D. Besides, elements of both lists are not single polynomials in $\mathcal{R}_q$, but pairs of polynomials. It was necessary to find a way to merge each pair into a single polynomial, as a separate treatment is easily attacked, for instance applying different permutations. Difficulties of this approach caused us to opt for an alternative way: Beneš networks.

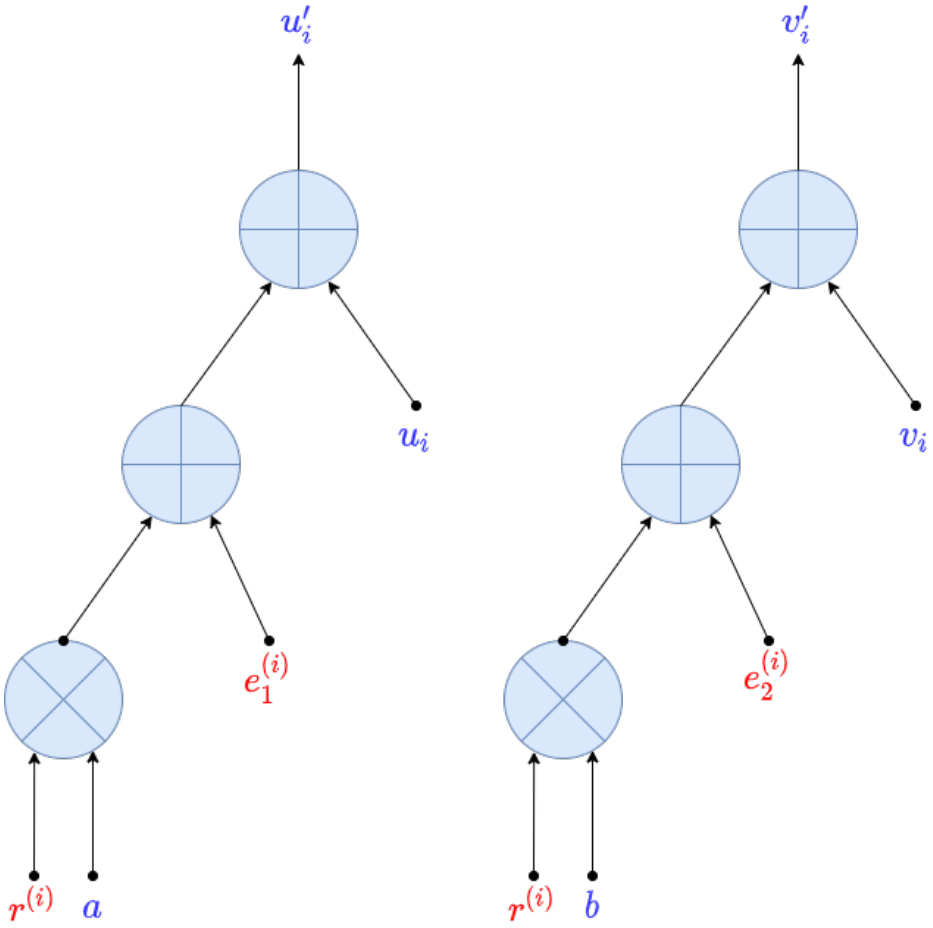

Figure 3: Arithmetic circuit for a RLWE re-encryption

6.2 The random permutation: Beneš networks

Now the next step is to perform the secret permutation. We will use as a model a permutation network called Beneš network proposed by Abraham Waksman in [Wak68]. The use of Beneš networks is not new in cryptography, as early results from Masayuki Abe [Abe99] already considered these constructions to apply them to mix-nets. Nevertheless, the asymptotic cost of these solutions were usually worse than others, and they were considered inefficient. In this case, the amortized cost of the proofs for Ajtai's function preimages and the satisfiability of arithmetic circuits may give a new opportunity to this kind of constructions.

Formally, a permutation network is an acyclic graph with N inputs and N outputs where vertices have in-degree and out-degree equal to 2. These vertices are called *switch gates* and each of them has a special input $b \in \{0, 1\}$, which indicates if the two inputs are switched or if they remain in the same order (see figure 4).

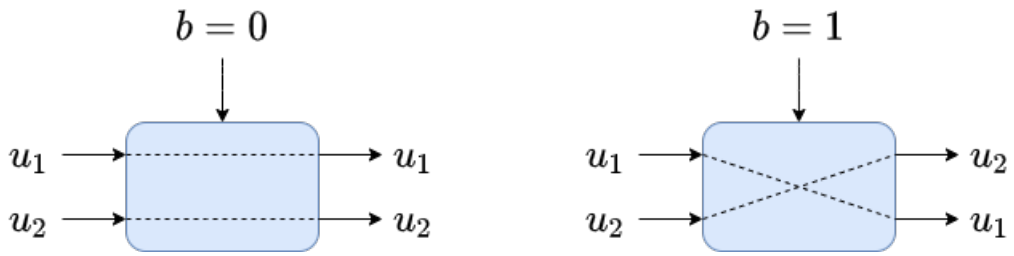

Figure 4: Switch gate

Consider a switch gate with inputs u_1 and u_2, and outputs v_1 and v_2. The switch consists in just applying the following operation:

$$v_1 = (1 - b) \cdot u_1 + b \cdot u_2$$
$$v_2 = (1 - b) \cdot u_2 + b \cdot u_1$$

Therefore, we can craft an arithmetic circuit to model this operation (figure 5).

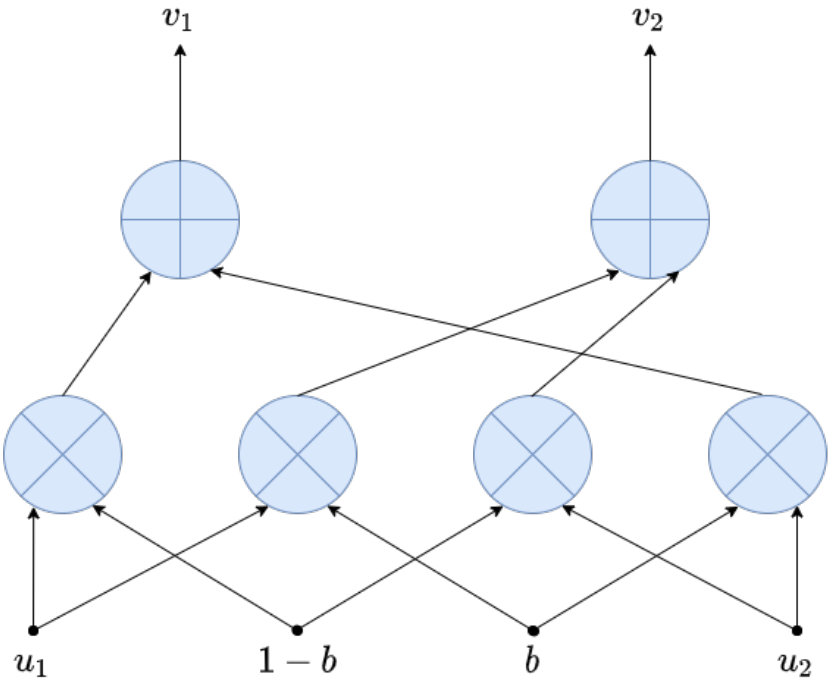

Figure 5: Arithmetic circuit for a switch gate

Notice that with this circuit nothing forces the wire $1 - b$ to actually be assigned to this value. In fact, the wire $1 - b$ is just an auxiliary value *aux* part of the witness that we can

state as the number which added to b results in 1, with an addition gate with inputs b and *aux* and output 1.

There is another issue regarding the switching decision bits. We are simply performing an arithmetic operation involving the inputs u_1, u_2, b and the outputs v_1, v_2. A malicious prover could replace the bit with whichever other polynomial in $\mathcal{R}_q$ and since the gate is expecting values in $\mathcal{R}_q$, it will still work, although not in the desirable way, as the outputs would be totally different from the inputs. If we were working over a field, we could just prove if $b(1-b) = 0$ holds, and that would imply that either $b = 0$ or $b = 1$. But the ring $\mathcal{R}_q$ is not integral domain, which means the existence of zero divisors, so this is not enough (although it will be useful). To sort out this issue we will use the following lemma:

Lemma 6.1 ([LN17]). *Let $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ where n is a power of 2 and q a prime such that $q \equiv 5 \mod 8$. Then every non-zero polynomial $a \in \mathcal{R}_q$ with $\|a\|_\infty < \sqrt{q/2}$ is invertible.*

In this case, if we prove that the euclidean norm of the bits is small, since the infinite norm is always smaller or equal than the euclidean norm, we prove that the bits are invertible, so they cannot be zero-divisors. Therefore if $b(1-b) = 0$, then the only possibilities are $b = 0$ or $b = 1$.

The bits must remain secret, since revealing them will reveal the secret permutation. We can take advantage of the zero knowledge proof 4.4.1 and theorem 4.14. We commit to the bits and prove knowledge of a valid opening with small norm for these commitments. It would remain to prove that $b(1-b) = 0$, but we can easily embed this proof in the circuit satisfiability proof. Recall that we also needed that $b + (1-b) = 1$, so we can give another sub-circuit in charge of checking consistency of the switch bits.

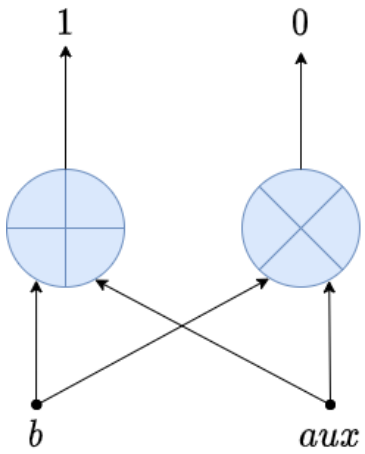

Figure 6: Arithmetic circuit for bit consistency

Now that it is clear how to implement the nodes of a permutation network using an arithmetic circuit, let us show how to combine them to craft a complete permutation network. One particularity of Beneš networks is that they are constructed recursively. A 2×2 Beneš network is just a switch gate, and it is trivial that a switch gate models every permutation of 2 elements, namely the identity if $b = 0$ and a switch if $b = 1$. Now we can construct a $2^k \times 2^k$ network using two $2^{k-1} \times 2^{k-1}$ Beneš sub-networks and 2^k switch gates, that will be able to perform whichever permutation of 2^k elements. See figure 7 for an example of an 8×8 network to clarify this construction.

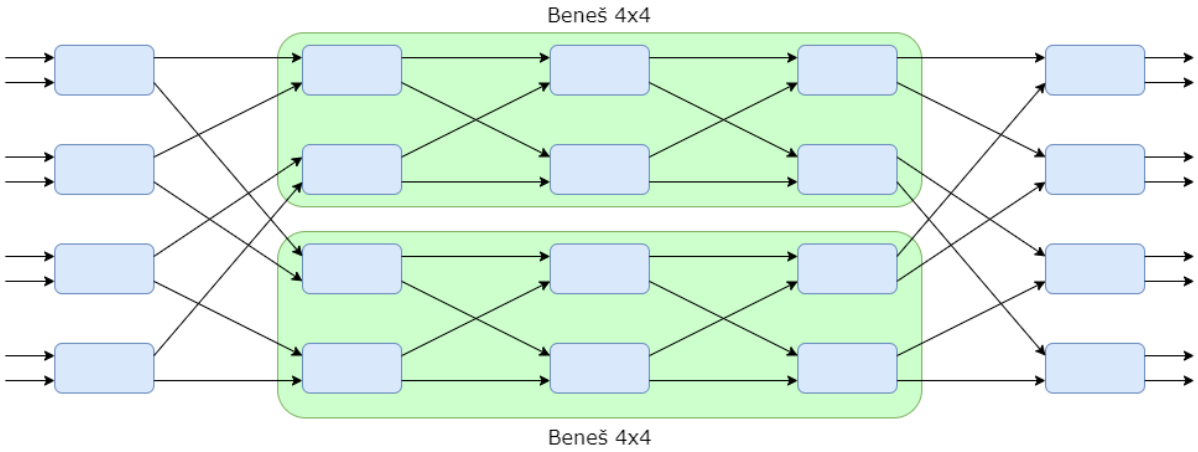

Figure 7: 8x8 Beneš Network

An easy induction yields that for $N = 2^k$, to craft a $N \times N$ network we will need $2\log_2(N) - 1$ stages of $N/2$ switch gates each, therefore an $\mathcal{O}(N \log(N))$ amount of switch gates. Unfortunately, this will worse the linear time complexity we achieved to perform the re-encryption. Nevertheless, the amortized proof for satisfiability given in section 5 still allows a $\mathcal{O}(\sqrt{N}\log^2(N))$ communication cost, which remains sub-linear in the number of voters. The decision to use Beneš networks is given because it is possible to easily model any of the $N!$ permutations without deadlocks (i.e. for each wire only travels one value). Besides, Beneš networks can be extended to arbitrary sizes, and not just powers of 2 [CM97].

One could imagine that to perform the permutation the prover could just choose the switch bit of each gate uniformly at random from $\{0, 1\}$, and let the circuit apply the resulting permutation. This will not be correct, since in a shuffle every permutation of N elements must have the same probability to appear. The random choosing of the bits implies that some permutation will appear more often than others, so the choice is not uniform, as shown in [AH01]. Therefore, if we denote by $\mathcal{S}_N$ the set of permutations of N elements, the prover must first choose $\pi \xleftarrow{\mathcal{R}} \mathcal{S}_N$, and then run an algorithm to set the bits accordingly. These algorithms are called *routing algorithms* and the best known have a complexity of $\mathcal{O}(N \log(N))$, such as [CCM09], so it does not affect the asymptotic complexity of the prover.

With these considerations, we can define a zero-knowledge argument to prove that a permutation was fixed in advance and applied to a list of elements of $\mathcal{R}_q$. The total amount of addition and multiplication gates for a switch gate is always 8, and for an $N \times N$ Beneš network we need $\mathcal{O}(N\log(N))$ switch gates, so the total size of the permutation circuit is $\mathcal{O}(N\log(N))$ on the number of votes.

Since we are working with RLWE encryption, whose ciphertexts are pairs of elements of $\mathcal{R}_q$, we must extend the calculations to pairs of elements, which can be easily done if we duplicate the amount of switch gates and carefully connect each bit to the two corresponding gates, one for each element of the pair. The amount of gates is doubled, so it does not affect the asymptotic cost of the circuit. With this construction we ensure that the permutation applied is the same for the first elements and the second ones.

$$ZKP\left[\ \pi \in \mathcal{S}_N \ \middle|\ \forall i \in [N] : (u''_{\pi(i)}, v''_{\pi(i)}) = (u'_i, v'_i)\ \right]$$

6.3 A new lattice-based protocol for the correctness of a shuffle

Finally we proceed to combine all the previous steps to provide a zero-knowledge argument for the correctness of a shuffle. What the proof allows to verify is that, given as statement a public list $C = \{c_1, \dots, c_N\}$, with $c_i = (u_i, v_i) \in \mathcal{R}_q^2$, another public list $D = \{d_1, \dots, d_N\}$, with $d_i = c''_i = (u''_i, v''_i) \in \mathcal{R}_q^2$, D is the result of shuffling C.

The prover witnesses are the random elements for the re-encryption $r^{(i)}, e_1^{(i)}, e_2^{(i)}$ for $i \in \{1, \dots, N\}$ and the permutation $\pi \in \mathcal{S}_N$.

$$ZKP\left[\ \begin{matrix}\pi \in \mathcal{S}_N \\ r^{(i)}, e_1^{(i)}, e_2^{(i)}, \quad i \in [N]\end{matrix}\ \middle|\ \forall i \in [N] : \left\{\begin{matrix}(u''_{\pi(i)}, v''_{\pi(i)}) = \text{Re-Enc}((u_i, v_i), r^{(i)}, e_1^{(i)}, e_2^{(i)}) \\ \left\|r^{(i)}\right\|_2, \left\|e_1^{(i)}\right\|_2, \left\|e_2^{(i)}\right\|_2 \leq \delta\end{matrix}\right.\ \right]$$

This proof will be equivalent to prove satisfiability for a circuit constructed following the sections above. The prover runs a routing algorithm to find the bits b_j for $j \in \{1, \dots, G\}$, where $G \in \mathcal{O}(N\log(N))$ is the number of switch gates that the circuit will require. The circuit has as public inputs conforming the statement the elements u_i, v_i for $i \in \{1, \dots, N\}$, and public outputs the elements u''_i, v''_i for $i \in \{1, \dots, N\}$. It also has the private inputs $r^{(i)}, e_1^{(i)}, e_2^{(i)}$ for $i \in \{1, \dots, N\}$ and the bits b_j for $j \in \{1, \dots, G\}$. Recall that for each bit the (honest) prover calculates $1 - b_j$ that will be also a private input aux_j whose value will be verified in the circuit.

Now, the overall protocol works as follows:

1. The prover and the verifier engage in a ZKP like 4.4.1 to prove $\left\|r^{(i)}\right\|_2, \left\|e_1^{(i)}\right\|_2, \left\|e_2^{(i)}\right\|_2 \leq \delta$, for a δ suitable considering the number of mix-nodes.
2. The prover and the verifier engage in a ZKP like 4.4.1 to prove $\|b_j\|_2 \leq \beta$, for a $\beta < \sqrt{q/2}$.
3. The prover and the verifier engage in a ZKP like the one explained in 5.3 applied to this particular circuit to prove its satisfiability.
4. The verifier accepts the proof if and only if all the previous proofs were accepted.

The overall complexity of the proof is dominated by the asymptotic cost of step 3, determined by the the total amount of addition and multiplication gates. There are $6N$ gates to perform the re-encryption and $2 \times 8G$ gates for the permutation. This yields an arithmetic circuit of $\mathcal{O}(N \log(N))$ gates. The complexity of the poof for a circuit with M gates was $\mathcal{O}(\sqrt{M \log^3(M)})$ in communication, $\mathcal{O}(M \log(M) \log^2(\lambda))$ for the prover and $\mathcal{O}(M \log^3(\lambda))$ for the verifier.

To calculate the complexity, since $M \in \mathcal{O}(N \log(N))$ and $\log(N \log(N)) \in \mathcal{O}(\log(N))$, we have a total complexity of:

- $\mathcal{O}(\sqrt{N} \log^2(N))$ for communication.
- $\mathcal{O}(N \log^2(N) \log^2(\lambda))$ for the prover.
- $\mathcal{O}(N \log(N) \log^3(\lambda))$ for the verifier.

Therefore, we achieve sub-linear communication cost in the overall argument, as we intended to.

7. Conclusions

We start this thesis introducing as mathematical background the main concepts about lattices and the main lattice-based problems such as the SIS or the RLWE, whose hardness work as a basis for the new post-quantum cryptography. This kind of cryptography is of capital importance nowadays due to the new paradigm of computation based on quantum physics, which is threatening the security of classical cryptography applications.

Then, we introduce the main cryptographic objects used in public-key cryptography, such as encryption, commitment schemes and zero-knowledge arguments. Moreover, we present some examples for each object, namely the RLWE encryption, a commitment scheme based on SIS and a zero-knowledge argument for openings of the previous commitment scheme.

The strategy to achieve the objective of the thesis, proving the correctness of a shuffle of RLWE ciphertexts, consists in explaining a protocol for general arithmetic circuit satisfiability and use this result for a certain circuit which models a shuffle. With this proof, an electronic voting is able to use mix-nets in order to provide anonymity to the process in a verifiable and secure way. Besides, the amortized complexity of the argument for circuits, alongside to the relatively small amount of gates of our circuit, yields an overall argument with sub-linear communication cost in time.

Post-quantum cryptography is a very recent branch of study for which new results appear almost constantly. For classical arguments based on the discrete logarithm assumption, there exist polylogarithmic time complexity arguments for the satisfiability of arithmetic circuits, while the argument we managed achieves an amortized complexity of fractional power. Recent research [BLNS20] shows the possibility to prove knowledge for openings of commitments with post-quantum security and in polylogarithmic communication complexity. This is an important improvement over the results in [BBC+18]. Therefore, by replacing [BBC+18] with [BLNS20] in our protocol, it might be possible to reduce the overall communication complexity from fractional power to polylogarithmic.

Nevertheless, the initial objective of this thesis has been met, since we have modeled the first post-quantum sub-linear protocol for the correctness of a shuffle.

References

[Abe99] Masayuki Abe, *Mix-networks on permutation networks*, Advances in Cryptology - ASIACRYPT'99 (Berlin, Heidelberg) (Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, eds.), Springer Berlin Heidelberg, 1999, pp. 258–273.

[AH01] Masayuki Abe and Fumitaka Hoshino, *Remarks on mix-network based on permutation networks*, Public Key Cryptography (Berlin, Heidelberg) (Kwangjo Kim, ed.), Springer Berlin Heidelberg, 2001, pp. 317–324.

[Ajt96] M. Ajtai, *Generating hard instances of lattice problems (extended abstract)*, Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '96, Association for Computing Machinery, 1996, p. 99–108.

[Ban93] Wojciech Banaszczyk, *New bounds in some transference theorems in the geometry of numbers*, Mathematische Annalen **296** (1993), 625–635.

[BBC+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafael del Pino, Jens Groth, and Vadim Lyubashevsky, *Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits*, Advances in Cryptology – CRYPTO 2018, Lecture Notes in Computer Science, vol. 10992, Springer, 2018, pp. 669–699.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit, *Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting*, Cryptology ePrint Archive, Report 2016/263, 2016, `https://eprint.iacr.org/2016/263`.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven, *Better zero-knowledge proofs for lattice encryption and their application to group signatures*, vol. 8873, 12 2014.

[BG12] Stephanie Bayer and Jens Groth, *Efficient zero-knowledge argument for correctness of a shuffle*, Advances in Cryptology – EUROCRYPT 2012 (Berlin, Heidelberg) (David Pointcheval and Thomas Johansson, eds.), Springer Berlin Heidelberg, 2012, pp. 263–280.

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler, *A non-pcp approach to succinct quantum-safe zero-knowledge*, Cryptology ePrint Archive, Report 2020/737, 2020, `https://eprint.iacr.org/2020/737`.

[CCM09] A. Chakrabarty, M. Collier, and S. Mukhopadhyay, *Matrix-based nonblocking routing algorithm for beneš networks*, 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009, pp. 551–556.

[CDK14] Ronald Cramer, Ivan Damgård, and Marcel Keller, *On the amortized complexity of zero-knowledge protocols*, Journal of Cryptology **27** (2014), no. 2, 284–316 (English).

[Cha81] David L. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, Commun. ACM **24** (1981), no. 2, 84–90.

[CM97] Chihming Chang and Rami Melhem, *Arbitrary size benes networks*, Parallel Processing Letters **07** (1997).

[CMM17] Núria Costa, Ramiro Martínez, and Paz Morillo, *Proof of a shuffle for lattice-based cryptography (full version)*, Cryptology ePrint Archive, Report 2017/900, 2017, `https://eprint.iacr.org/2017/900`.

[DH76] Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, IEEE Trans. Inf. Theory **22** (1976), 644–654.

[FS87] Amos Fiat and Adi Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, Advances in Cryptology — CRYPTO' 86 (Berlin, Heidelberg) (Andrew M. Odlyzko, ed.), Springer Berlin Heidelberg, 1987, pp. 186–194.

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Pierre Seifert, *Approximating shortest lattice vectors is not harder than approximating closest lattice vectors*, Inf. Process. Lett. **71** (1999), 55–61.

[GN08] Nicolas Gama and Phong Q. Nguyen, *Finding short lattice vectors within mordell's inequality*, Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '08, Association for Computing Machinery, 2008, p. 207–216.

[Gro09] Jens Groth, *Linear algebra with sub-linear zero-knowledge arguments*, 08 2009, pp. 192–208.

[HILL99] Johan Hastad, Russell Impagliazzo, Leonid Levin, and Michael Luby, *A pseudorandom generator from any one-way function*, SIAM Journal on Computing **28** (1999).

[LLL82] Arjen Lenstra, H. Lenstra, and László Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen **261** (1982).

[LN17] Vadim Lyubashevsky and Gregory Neven, *One-shot verifiable encryption from lattices*, Cryptology ePrint Archive, Report 2017/122, 2017, `https://eprint.iacr.org/2017/122`.

[LP10] Richard Lindner and Chris Peikert, *Better key sizes (and attacks) for lwe-based encryption*, Cryptology ePrint Archive, Report 2010/613, 2010, `https://eprint.iacr.org/2010/613`.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev, *On ideal lattices and learning with errors over rings*, J. ACM **60** (2013), no. 6.

[Lyu12] Vadim Lyubashevsky, *Lattice signatures without trapdoors*, Advances in Cryptology – EUROCRYPT 2012 (Berlin, Heidelberg) (David Pointcheval and Thomas Johansson, eds.), Springer Berlin Heidelberg, 2012, pp. 738–755.

[Mar18] Ramiro Martínez Pinilla, *Fully post-quantum protocols for e-voting, coercion resistant cast as intended and mixing networks*, Master's thesis, Universitat Politécnica de Catalunya, January 2018.

[MR09] Daniele Micciancio and Oded Regev, *Lattice-based cryptography*, pp. 147–191, 01 2009.

[Reg05] Oded Regev, *On lattices, learning with errors, random linear codes, and cryptography*, In STOC, ACM Press, 2005, pp. 84–93.

[RS10] Markus Rückert and Michael Schneider, *Estimating the security of lattice-based cryptosystems*, Cryptology ePrint Archive, Report 2010/137, 2010, `https://eprint.iacr.org/2010/137`.

[San16] Eduard Sanou Gozalo, *Post-quantum cryptography: Lattice-based encryption*, Master's thesis, Universitat Politécnica de Catalunya, July 2016.

[Sho97] Peter W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26** (1997), no. 5, 1484–1509.

[Wak68] Abraham Waksman, *A permutation network*, J. ACM **15** (1968), no. 1, 159–163.

A. Argument for multiplicative relations

$$ZKP\left[A_i, B_i, C_i \in \mathbb{Z}_q^{k\times n} \;\middle|\; \forall i \in \{1,\dots,m\} : A_i \circ B_i \equiv C_i \mod q \right]$$

Protocol 2: Proof for multiplicative relations

$\mathcal{P}(A_i, B_i, C_i, ck, \mathcal{B})$	$\mathcal{V}(ck, \mathcal{B})$
$\begin{bmatrix} A_i \\ \mathbf{0}^{k\times n} \end{bmatrix} \bigcirc \begin{bmatrix} B_i \\ \mathbf{0}^{k\times n} \end{bmatrix} = \begin{bmatrix} C_i \\ C_i' \end{bmatrix} \mod q$	
$A_0, B_{m+1} \xleftarrow{\mathcal{R}} D_{\sigma_1}^{2k\times n}$	
For $1 \le i \le m$:	
$\alpha_i, \beta_i \xleftarrow{\mathcal{R}} [q]^{k\times n'}$	
$\gamma_i \xleftarrow{\mathcal{R}} [q]^{2k\times n'}$	
$\mathbf{A}_i = \mathrm{Com}_{ck}^*\left(\begin{bmatrix} A_i \\ \mathbf{0}^{k\times n} \end{bmatrix}; \begin{bmatrix} \alpha_i \\ \mathbf{0}^{k\times n'} \end{bmatrix}\right)$	
$\mathbf{B}_i = \mathrm{Com}_{ck}^*\left(\begin{bmatrix} B_i \\ \mathbf{0}^{k\times n} \end{bmatrix}; \begin{bmatrix} \beta_i \\ \mathbf{0}^{k\times n'} \end{bmatrix}\right)$	
$\mathbf{C}_i = \mathrm{Com}_{ck}^*\left(\begin{bmatrix} C_i \\ C_i' \end{bmatrix}; \gamma_i\right)$	
$\alpha_0, \beta_{m+1} \xleftarrow{\mathcal{R}} D_{\sigma_1}^{2k\times n'}$	
$\mathbf{A}_0 = \mathrm{Com}_{ck}^*(A_0; \alpha_0)$	
$\mathbf{B}_{m+1} = \mathrm{Com}_{ck}^*(B_{m+1}; \beta_{m+1})$	
$\xrightarrow{\{\mathbf{A}_i\}_{i=0}^m, \{\mathbf{B}_i\}_{i=1}^{m+1}, \{\mathbf{C}_i\}_{i=1}^m}$	
	$\mathbf{y} \xleftarrow{\mathcal{R}} [q]^{2k}$
$\xleftarrow{\mathbf{y}}$	
$A(\mathbf{X}) = A_0 + \sum_{i=1}^m M_{\mathbf{X}}^i (M_{\mathbf{y}}^i \mod q) \begin{bmatrix} A_i \\ \mathbf{0}^{k\times n} \end{bmatrix}$	
$B(\mathbf{X}) = B_{m+1} + \sum_{i=1}^m M_{\mathbf{X}}^{m+1-i} \begin{bmatrix} B_i \\ \mathbf{0}^{k\times n} \end{bmatrix}$	

$$C = \sum_{i=1}^{m} M_{\mathbf{y}}^{i} \begin{bmatrix} C_i \\ C_i' \end{bmatrix} \mod q$$

$$A(\mathbf{X}) \bigcirc B(\mathbf{X}) \mod q = M_{\mathbf{X}}^{m+1} C + \sum_{\substack{l=0 \\ l \neq m+1}}^{2m} M_{\mathbf{X}}^{l} H_l \mod q$$

For $0 \leq l \leq 2m$, $l \neq m+1$:

$\eta_l \xleftarrow{\mathcal{R}} [q]^{2k \times n'}$

$\mathbf{H}_l = \mathsf{Com}_{ck}^{*}(H_l; \eta_l)$

$$\xrightarrow{\{\mathbf{H}_l\}_{l=0, l \neq m}^{2m}}$$

$\mathbf{x} \xleftarrow{\mathcal{R}} [q]^{2k}$

$$\xleftarrow{\mathbf{x}}$$

$$A = A_0 + \sum_{i=1}^{m} M_{\mathbf{x}}^{i} (M_{\mathbf{y}}^{i} \mod q) \begin{bmatrix} A_i \\ \mathbf{0}^{k \times n} \end{bmatrix}$$

$$\alpha = \alpha_0 + \sum_{i=1}^{m} M_{\mathbf{x}}^{i} (M_{\mathbf{y}}^{i} \mod q) \begin{bmatrix} \alpha_i \\ \mathbf{0}^{k \times n'} \end{bmatrix}$$

$$B = B_{m+1} + \sum_{i=1}^{m} (M_{\mathbf{x}}^{m+1-i} \mod q) \begin{bmatrix} B_i \\ \mathbf{0}^{k \times n} \end{bmatrix}$$

$$\beta = \beta_{m+1} + \sum_{i=1}^{m} (M_{\mathbf{x}}^{m+1-i} \mod q) \begin{bmatrix} \beta_i \\ \mathbf{0}^{k \times n'} \end{bmatrix}$$

$$D = (A \bigcirc B \mod q) - \sum_{i=1}^{m} (M_{\mathbf{y}}^{i} \mod q) \begin{bmatrix} C_i \\ C_i' \end{bmatrix} - \sum_{\substack{l=0 \\ l \neq m+1}}^{2m} (M_{\mathbf{x}}^{l} \mod q) H_l$$

$\delta \xleftarrow{\mathcal{R}} D_{\sigma_2}^{2k \times n'}$

$\mathbf{D} = \mathsf{Com}_{ck}^{*}(D; \delta)$

$E \xleftarrow{\mathcal{R}} p \cdot D_{\sigma_3}^{2k \times n}$

$\epsilon \xleftarrow{\mathcal{R}} D_{\sigma_4}^{2k \times n'}$

$\mathbf{E} = \mathsf{Com}_{ck}^{*}(E; \epsilon)$

$$\xrightarrow{\mathbf{D}, \mathbf{E}}$$

$\mathbf{z} \xleftarrow{\mathcal{R}} [q]^{2k}$

$$\xleftarrow{\mathbf{z}}$$

$\mathcal{P}(A_i, B_i, C_i, ck, \mathcal{B})$	$\mathcal{V}(ck, \mathcal{B})$

$\text{Rej}((A\|\alpha\|B\|\beta), (A\|\alpha\|B\|\beta) - (A_0\|\alpha_0\|B_{m+1}\|\beta_{m+1}), \sigma_1, e)$

$\rho = \sum_{i=1}^{m}(M_{\mathbf{x}}^{m+1}M_{\mathbf{y}}^{i} \mod q)\gamma_i + \sum_{\substack{l=0 \\ l\neq m+1}}^{2m}(M_{\mathbf{x}}^{l} \mod q)\eta_l + \delta$

$\text{Rej}(\rho, \rho - \delta, \sigma_2, e)$

$\bar{D} = (M_{\mathbf{z}} \mod q)D + E$

$\bar{\delta} = (M_{\mathbf{z}} \mod q)\delta + e$

$\text{Rej}(\bar{D}/q, D/q, \sigma_3, e)$

$\text{Rej}(\bar{\delta}, \delta, \sigma_4, e)$

$$\xrightarrow{A, \alpha, B, \beta, \rho, \bar{D}, \bar{\delta}}$$

$\mathcal{P}$ and $\mathcal{V}$ engage in a ZKP like 4.4.1 including every commitment sent from $\mathcal{P}$ to $\mathcal{V}$

$$\text{Com}_{ck}^{*}(A, \alpha) \stackrel{?}{=} \sum_{i=0}^{m}(M_{\mathbf{x}}^{i}M_{\mathbf{y}}^{i} \mod q)\mathbf{A}_i$$

$$\text{Com}_{ck}^{*}(B, \beta) \stackrel{?}{=} \sum_{i=1}^{m+1}(M_{\mathbf{x}}^{m+1-i} \mod q)\mathbf{B}_i$$

$$\text{Com}_{ck}^{*}(A \bigcirc B \mod q, \rho) \stackrel{?}{=} \sum_{i=1}^{m}(M_{\mathbf{x}}^{m+1}M_{\mathbf{y}}^{i} \mod q)\mathbf{C}_i + \sum_{\substack{l=0 \\ l\neq m+1}}^{2m}(M_{\mathbf{x}}^{l} \mod q)\mathbf{H}_l + \mathbf{D}$$

$$\text{Com}_{ck}^{*}(\bar{D}, \bar{\delta}) \stackrel{?}{=} M_{\mathbf{z}} \mod q)\mathbf{D} + \mathbf{E}$$

$$\bar{D} \stackrel{?}{=} 0 \mod q$$

$$\left\|\bar{D}\right\|_2 \stackrel{?}{\leq} 2\sqrt{kn}\sigma_3 q$$

$$\|(A\|\alpha\|B\|\beta)\|_2 \stackrel{?}{\leq} 4\sqrt{kn}\sigma_1$$

$$\|\rho\|_2 \stackrel{?}{\leq} 2\sqrt{kn}\sigma_2$$

$$\left\|\bar{\delta}\right\|_2 \stackrel{?}{\leq} 2\sqrt{kn}\sigma_4$$

Standard deviations:

$$\sigma_1 = 48\sqrt{kn}kmq^2 \qquad \sigma_2 = 72\sqrt{2kn}kmq$$
$$\sigma_3 = 24\sqrt{2kn}kq(1+6kmq) \qquad \sigma_4 = 24\sqrt{2}k^2qn\sigma_2$$

B. Argument for linear consistency constraints

$$ZKP\left[\ \mathbf{a}_{i,j},\ \mathbf{b}_{i,j},\ \mathbf{c}_{i,j} \in \mathbb{Z}_q^n \ \middle|\ \sum_{i=1,j=1}^{m,k} \mathbf{a_{i,j}} \cdot \mathbf{w}_{u,a,i,j} + \sum_{i=1,j=1}^{m,k} \mathbf{b_{i,j}} \cdot \mathbf{w}_{u,b,i,j} + \sum_{i=1,j=1}^{m,k} \mathbf{c_{i,j}} \cdot \mathbf{w}_{u,c,i,j} = K_u \right]$$

Protocol 3: Proof for linear consistency constraints

$\mathcal{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}, \mathbf{c}_{i,j}, ck, \mathcal{B}')$	$\mathcal{V}(ck, \mathcal{B}')$
$A_0, B_0, C_0 \xleftarrow{\mathcal{R}} D_{\sigma_1}^{2k \times n}$	
$\alpha_0, \beta_0, \gamma_0 \xleftarrow{\mathcal{R}} D_{\sigma_1}^{2k \times n'}$	
$\mathbf{A}_0 = \mathrm{Com}_{ck}^*(A_0, \alpha_0)$	
$\mathbf{B}_0 = \mathrm{Com}_{ck}^*(B_0, \beta_0)$	
$\mathbf{C}_0 = \mathrm{Com}_{ck}^*(C_0, \gamma_0)$	
For $1 \leq i \leq m$:	
$\mathcal{P}$ arrange $\mathbf{a}_{i,j}, \mathbf{b}_{i,j}, \mathbf{c}_{i,j}$ into matrices $A_i, B_i, C_i \in [q]^{k \times n}$	
$\alpha_i, \beta_i, \gamma_i \xleftarrow{\mathcal{R}} [q]^{k \times n'}$	
$\mathbf{A}_i = \mathrm{Com}_{ck}^*\left(\begin{bmatrix} A_i \\ \mathbf{0}^{k \times n} \end{bmatrix}; \begin{bmatrix} \alpha_i \\ \mathbf{0}^{k \times n'} \end{bmatrix}\right)$	
$\mathbf{B}_i = \mathrm{Com}_{ck}^*\left(\begin{bmatrix} B_i \\ \mathbf{0}^{k \times n} \end{bmatrix}; \begin{bmatrix} \beta_i \\ \mathbf{0}^{k \times n'} \end{bmatrix}\right)$	
$\mathbf{C}_i = \mathrm{Com}_{ck}^*\left(\begin{bmatrix} C_i \\ \mathbf{0}^{k \times n} \end{bmatrix}; \begin{bmatrix} \gamma_i \\ \mathbf{0}^{k \times n'} \end{bmatrix}\right)$	
$\xrightarrow{\{\mathbf{A}_i\}_{i=0}^m, \{\mathbf{B}_i\}_{i=0}^m, \{\mathbf{C}_i\}_{i=0}^m}$	
	$\mathbf{y} \xleftarrow{\mathcal{R}} [q]^{2k}$
$\xleftarrow{\mathbf{y}}$	
Compute $a(\mathbf{X}), b(\mathbf{X}), c(\mathbf{X})$	

$\mathcal{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}, \mathbf{c}_{i,j}, ck, \mathcal{B}')$ $\mathcal{V}(ck, \mathcal{B}')$

$$A(\mathbf{X}) = A_0 + \sum_{i=1}^{m} M_{\mathbf{X}}^i \begin{bmatrix} A_i \\ \mathbf{0}^{k \times n} \end{bmatrix}$$
$$B(\mathbf{X}) = B_0 + \sum_{i=1}^{m} M_{\mathbf{X}}^i \begin{bmatrix} B_i \\ \mathbf{0}^{k \times n} \end{bmatrix}$$
$$C(\mathbf{X}) = C_0 + \sum_{i=1}^{m} M_{\mathbf{X}}^i \begin{bmatrix} C_i \\ \mathbf{0}^{k \times n} \end{bmatrix}$$

Compute for a (and analogously for b and c):

$$W_{u,a,i} = \begin{pmatrix} \mathbf{w}_{u,a,i,k} \\ \mathbf{w}_{u,a,i,k-1} \\ \vdots \\ \mathbf{w}_{u,a,i,1} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \in [q]^{2k \times n}$$

$$W_{a,i} = \sum_{u=1}^{U} (M_{\mathbf{y}}^u \mod q) W_{u,a,i}$$

$$W_a(\mathbf{X}) = \sum_{i=1}^{m} (M_{\mathbf{X}}^{m+1-i} \mod q) W_{a,i}$$

$$a(\mathbf{X}) \bigodot w_a + b(\mathbf{X}) \bigodot w_b + c(\mathbf{X}) \bigodot w_c \mod q =$$

$$= M_{\mathbf{X}}^{m+1} \sum_{u=1}^{U} (M_{\mathbf{y}}^u \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_u \\ 0 \\ \vdots \\ 0 \end{pmatrix} \mod q + M_{\mathbf{X}}^{m+1} \begin{pmatrix} k'_1 \\ \vdots \\ k'_{k-1} \\ 0 \\ k'_{k+1} \\ \vdots \\ k'_{2k} \end{pmatrix} + \sum_{\substack{l=0 \\ l \neq m+1}}^{2m} M_{\mathbf{X}}^l H_l \mod q$$

Write $K' = \begin{pmatrix} k'_1 \\ \vdots \\ k'_{k-1} \\ 0 \\ k'_{k+1} \\ \vdots \\ k'_{2k} \end{pmatrix}$

$\mathcal{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}, \mathbf{c}_{i,j}, ck, \mathcal{B}')$ $\qquad$ $\mathcal{V}(ck, \mathcal{B}')$

For $0 \leq l \leq 2m$, $l \neq m+1$:

$\eta_l \xleftarrow{\mathcal{R}} [q]^{2k \times n'}$

$\mathbf{H}_l = \mathsf{Com}^*_{ck}(H_l; \eta_l)$

$\kappa' \in [q]^{2k \times n'}$, with columns uniformly from $[q]^n$ and setting the kth row to zero

$\mathbf{K}' = \mathsf{Com}^*_{ck}(K', \kappa')$

$\mathbf{K}', \{\mathbf{H}_l\}_{l=0, l \neq m+1}^{2m} \longrightarrow$

$\mathbf{x} \xleftarrow{\mathcal{R}} [q]^{2k}$

$\leftarrow \mathbf{x}$

$$A = A_0 + \sum_{i=1}^{m} M^i_{\mathbf{x}} \begin{bmatrix} A_i \\ \mathbf{0}^{k \times n} \end{bmatrix} \mod q \ (B,\ C \text{ analogously})$$

$$\alpha = \alpha_0 + \sum_{i=1}^{m} M^i_{\mathbf{x}} \begin{bmatrix} \alpha_i \\ \mathbf{0}^{k \times n'} \end{bmatrix} \mod q \ (\beta,\ \gamma \text{ analogously})$$

$$W_a = \sum_{i=1}^{m} M^{m+1-i}_{\mathbf{x}} W_{a,i} \mod q \ (W_b,\ W_c \text{ analogously})$$

$$D = (A \bigodot W_a + B \bigodot W_b + C \bigodot W_c \mod q) -$$

$$- \sum_{u=1}^{U} (M^{m+1}_{\mathbf{x}} M^u_{\mathbf{y}}) \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_u \\ 0 \\ \vdots \\ 0 \end{pmatrix} \mod q - (M^{m+1}_{\mathbf{x}} \mod q) K' - \sum_{\substack{l=0 \\ l \neq m+1}}^{2m} (M^l_{\mathbf{x}} \mod q) H_l$$

$\delta \xleftarrow{\mathcal{R}} D^{2k \times n'}_{\sigma_2}$

$\mathbf{D} = \mathsf{Com}^*_{ck}(D; \delta)$

$E \xleftarrow{\mathcal{R}} q \cdot D^{2k \times n}_{\sigma_3}$

$\epsilon \xleftarrow{\mathcal{R}} D^{2k \times n'}_{\sigma_4}$

$\mathbf{E} = \mathsf{Com}^*_{ck}(E; \epsilon)$

$$\rho = (M^{m+1}_{\mathbf{x}} \mod q) \kappa + \sum_{\substack{l=0 \\ l \neq m+1}}^{2m} (M^l_{\mathbf{x}} \mod q) \eta_l + \delta$$

$\mathcal{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}, \mathbf{c}_{i,j}, ck, \mathcal{B}')$ $\mathcal{V}(ck, \mathcal{B}')$

$\mathbf{D}, \mathbf{E}$ $\longrightarrow$

$\mathbf{z} \xleftarrow{\mathcal{R}} [q]^{2k}$

$\mathbf{z}$ $\longleftarrow$

$\bar{D} = (M_{\mathbf{z}} \mod q)D + E$

$\bar{\delta} = (M_{\mathbf{z}} \mod q)\delta + e$

$\text{Rej}(\bar{D}/q, D/q, \sigma_3, e)$

$\text{Rej}(\bar{\delta}, \delta, \sigma_2, e)$

$\text{Rej}((A\|\alpha\|B\|\beta\|C\|\gamma), (A\|\alpha\|B\|\beta\|C\|\gamma) - (A_0\|\alpha_0\|B_0\|\beta_0\|C_0\|\gamma_0), \sigma_1, e)$

$A, \alpha, B, \beta, C, \gamma, \rho, \bar{D}, \bar{\delta}$ $\longrightarrow$

$\mathcal{P}$ and $\mathcal{V}$ engage in a ZKP like 4.4.1 including every commitment sent from $\mathcal{P}$ to $\mathcal{V}$

$$\mathbf{A} \stackrel{?}{=} \sum_{i=0}^{m} (M_{\mathbf{x}}^i \mod q)\mathbf{A}_i$$

$$\mathbf{B} \stackrel{?}{=} \sum_{i=0}^{m} (M_{\mathbf{x}}^i \mod q)\mathbf{B}_i$$

$$\mathbf{C} \stackrel{?}{=} \sum_{i=0}^{m} (M_{\mathbf{x}}^i \mod q)\mathbf{C}_i$$

$$\text{Com}_{ck}^*(A \bigodot W_a + B \bigodot W_b + C \bigodot W_c \mod q; \rho) \stackrel{?}{=}$$

$$\stackrel{?}{=} \sum_{u=1}^{U} M_{\mathbf{x}}^{m+1} M_{\mathbf{y}}^u \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \text{Com}_{ck}^*(K_u; \mathbf{0}) \\ 0 \\ \vdots \\ 0 \end{pmatrix} \mod q +$$

$$+ (M_{\mathbf{x}}^{m+1} \mod q)\mathbf{K}' + \sum_{\substack{l=0 \\ l \neq m+1}}^{2m} (M_{\mathbf{x}}^l \mod q)H_l$$

$\mathcal{P}(A_i, B_i, C_i, ck, \mathcal{B})$	$\mathcal{V}(ck, \mathcal{B})$
	$\bar{\mathbf{D}} \stackrel{?}{=} (M_{\mathbf{z}} \mod q)\mathbf{D} + \mathbf{E}$
	$\bar{D} \stackrel{?}{=} 0 \mod q$
	$\|(A\|\alpha\|B\|\beta\|C\|\gamma)\|_2 \stackrel{?}{\leq} 2\sqrt{6kn}\sigma_1$
	$\left\|\bar{D}\right\|_2 \stackrel{?}{\leq} 2\sqrt{k}\sigma_3 q$
	$\|\rho\|_2 \stackrel{?}{\leq} 2\sqrt{k}\sigma_2$
	$\left\|\bar{\delta}\right\|_2 \stackrel{?}{\leq} 2\sqrt{k}\sigma_4$

Standard deviations:

$$\sigma_1 = 48\sqrt{kn}kmq^2 \qquad \sigma_2 = 24\sqrt{2kn}kq^2(2m+1)$$
$$\sigma_3 = 24\sqrt{2kn}(1+k+2mkq) \qquad \sigma_4 = 24\sqrt{2kn}(2m+1)kq^2$$